Vysoká škola báňská - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Ovládání kamery BASLER A631f grafickým uživatelským rozhraním v softwareovém prostředí MATLAB

Controlling the Camera Basler A631f through GUI in MATLAB

2013 **Filip Volný**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Filip Volný**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Ovládání kamery BASLER A631f grafickým uživatelským rozhraním v softwareovém prostředí MATLAB
Controlling the Camera Basler A631f through GUI in MATLAB

Zásady pro vypracování:

1. Popis komunikace kamery BASLER A631f s prostředím MATLAB.
2. Vytvoření GUI pro práci s kamerou v MATLABu.
3. Vytvoření manuálu pro práci s GUI.

Seznam doporučené odborné literatury:

[1] ZAPLATÍLEK, K.; DOŇAR, B. *MATLAB: tvorba uživatelských aplikací.* Vydání 1. Praha: BEN - technická literatura, 2004. 215 s. ISBN 80-7300-133-0.

[2] Kovářík, Martin. *Programování a tvorba grafiky v Matlabu.* Vyd. 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2008. 130 s. : il. ISBN 978-80-7318-754-5 (brož.).

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 18.11.2011

Datum odevzdání: 07.05.2013

prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 06.05.2013

Filip Volný

.....

podpis studenta

Poděkování

Rád bych poděkoval svému vedoucímu, Ing. Janu Skapovi, Ph.D., za cenné rady, odbornou pomoc, podnětné konzultace a vstřícný přístup při vytváření této bakalářské práce. Dále bych chtěl poděkovat svým rodičům za podporu a vytvoření klidného zázemí pro tvorbu této bakalářské práce.

Abstrakt

Cílem této bakalářské práce je vytvořit grafické uživatelské rozhraní v programu MATLAB, které bude sloužit k ovládání digitální kamery Basler A631f. V jednotlivých kapitolách se budu zabývat samotnou kamerou, jejími specifikacemi, vlastnostmi a propojením s PC. Dále přiblížím program MATLAB a jeho součásti, které využívám, a Measurement & Automation Explorer. Nakonec popíši mnou vytvořené GUI a jeho zdrojový kód.

Klíčová slova

MATLAB, GUI, GUIDE, Basler, A631f, kamera, MAX, Measurement & Automation Explorer, Imaqtool, Image Acqusition toolbox

Abstract

The objective of this bachelor thesis is to create graphical user interface in MATLAB program, which will be used for controlling digital camera Basler A631f. In individual chapters I describe the camera itself, its specifications, properties and link with PC. Then I focus close on MATLAB program and its components, which I use, and Measurement & Automation Explorer. In the end I describe GUI created by myself and its source code.

Key words

MATLAB, GUI, GUIDE, Basler, A631f, camera, MAX, Measurement & Automation Explorer, Imaqtool, Image Acqusition Toolbox

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
DCAM	Digital Camcorder	Digitální Kamera
FPS	Frames per Second	Frekvence snímkování
FPT	Frames per Trigger	Počet snímků na jeden trigger
GUI	Graphical User Interface	Grafické uživatelské prostředí
GUIDE	Graphical User Interface Development Environment	Vývojové prostředí pro vytváření GUI
IEEE	Institute of Electrical and Electronics Engineers	Institut pro elektrotechnické a elektronické inženýrství
Imaqtool	Image Acquisition Toolbox	Sada nástrojů pro práci s videorekordéry
LSB	Least Significant bit	Nejméně významný bit
MSB	Most Significant bit	Nejvýznamnější bit
PC	Personal Computer	Osobní počítač
RJ-45	Registered Jack 45	Označení pro typ konektoru
RJ-50	Registered Jack 50	Označení pro typ konektoru
ROI	Region of Interest	Výřez snímku, která nás zajímá více než okolí

Seznam cizích použitých termínů

Termín	Význam termínu
Acquisition	Proces snímání kamerou
Big Endian	Způsob uložení dat do paměti
Callback	Zpětná vazba pro GUI komponenty v programu MATLAB
Driver	Ovladač
Edit Text Box	Textové pole
Figure	Grafické okno aplikací napsaných v programu MATLAB
FireWire	Sériová sběrnice pro připojení periférií k počítači
Frame	Snímek
Little Endian	Způsob uložení dat do paměti
Pin	Vývod elektronické součástky
Pixel	Obrazový bod
Preview	Náhled snímku
Push button	Tlačítko
Radio button	Výběrové tlačítko
Slider	Posuvník
Socket	Zdířka
Tag	Název GUI komponenty
Toggle Button	Přepínací tlačítko
Toolbox	Sada nástrojů vývojového prostředí
Trigger	Spouštěč snímání kamery
Updater	Malý program, který umožňuje aktualizaci

Obsah

Úvod

Téměř každý z nás již někdy pracoval s nějakou aplikací, aby mohl ovládat přístroje nebo programy. To, v čem běžně pracujeme, se nazývá grafické uživatelské rozhraní (GUI). Ne každé GUI však je pro uživatele příjemné a intuitivní. Cílem mé bakalářské práce je vytvořit GUI pro kameru Basler A631f, které bude splňovat standardní kritéria pro tvorbu grafických rozhraní.

V kapitole číslo 1 vás seznámím se samotnou kamerou. Popíši její základní vlastnosti a specifikace a přiblížím vám její vzhled. Nakonec krok po kroku ukážu, jak se propojuje s PC a jak lze otestovat její funkčnost.

V 2. kapitole budu popisovat program MATLAB, protože právě jej využívám k tvorbě svého GUI. Uvedu o něm základní informace a popíši jeho vzhled a vlastnosti. V další části této kapitoly se věnuji součástem MATLAB-Image Acquisiton Toolboxu a GUIDE. Image Acquisition Toolbox využívám jako zdroj informací a částečně také jako inspiraci pro tvorbu GUI. GUIDE je komponenta, ve které tvořím grafickou podobu GUI. Nakonec se budu okrajově zabývat problematikou tvorby grafických prostředí.

Ve 3. kapitole se budu zabývat softwarem Measurement & Automation Explorer. Na základě ovládací aplikace pro kameru Basler, jak ji vyobrazuje tento program, jsem vytvořil i své GUI. Výsledná podoba však bude samozřejmě odlišná.

Poslední kapitola bude sloužit jako manuál a dokumentace pro výslednou aplikaci. Kompletně rozeberu vzhled a funkcionalitu svého GUI a zmíním se o propojeních a interakcích mezi jednotlivými komponentami.

1 Kamera Basler A631f

1.1 Základní informace o kameře

Kamera Basler A631f je digitální monochromatická kamera. Maximální rozlišení, které poskytuje, je 1392 x 1040 obrazových bodů. Všechna podporovaná rozlišení a jejich frekvence snímkování (fps) jsou rozepsány v tabulce 1. Snímkovací frekvence u maximálního rozlišení je řešena jiným nastavením než u ostatních rozlišení, viz kapitola 3.2.1 vzorec 1.

Kamera dále nabízí volbu mezi třemi výstupními video formáty. Jedná se o Mono 8, Mono 16 a Pseudo YUV 4:2:2. Mono 8 udává, že se přenáší 8 bitů na pixel.

Mono 16 přenáší 16 bitů na pixel. Výstupem sice je 16 bitů na pixel, ale pouze 12 jich je efektivních. To znamená, že pouze 12 bitů přenáší informace o daném pixelu a zbylé 4 bity jsou naplněny nulami. Bity jsou plněny od nejméně významného bitu (LSB), přičemž data jsou do paměti počítače ukládána ve formátu Little Endian. Na paměťové místo s nejnižší adresou se tedy uloží LSB a za něj se ukládají další bity až po nejvíce významný bit (MSB).

Formát Pseudo YUV 4:2:2 je zde lehce výjimečný, protože obvykle bývá spjat s barevnými kamerami, avšak u typu A631f je také dostupný. Přestože se jedná o barevný formát, bude zde výstup černobílý, jelikož se jedná o monochromní kameru. Tento formát je podporovaný hlavně z toho důvodu, aby naše kamera mohla být používána v systémových doplňcích Windows XP, jako je například Movie Maker. [1]

Tabulka 1: Dostupné fps pro jednotlivá rozlišení

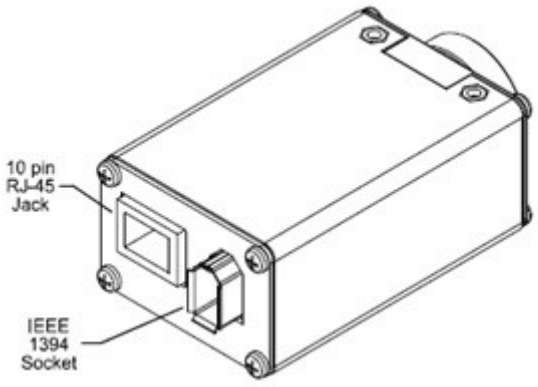
Rozlišení, Formát	Snímková frekvence (fps)				
	1,875	3,75	7,5	15	30
320 x 240 YUV 4:2:2	x	x	x	x	x
640 x 480 Mono 8	x	x	x	x	x
800 x 600 Mono 8			x	x	
1024 x 768 Mono 8		x	x	x	
1280 x 960 Mono 8	x	x	x	x	
640 x 480 Mono 16	x	x	x	x	x
800 x 600 Mono 16		x	x	x	
1024 x 768 Mono 16		x	x	x	
1280 x 960 Mono 16	x	x	x		
1392 x 1040 Mono 8	hodnota fps viz kapitola 3.2.1 vzorec 1				
1392 x 1040 Mono 16	hodnota fps viz kapitola 3.2.1 vzorec 1				

1.2 Vzhled a propojení kamery s PC

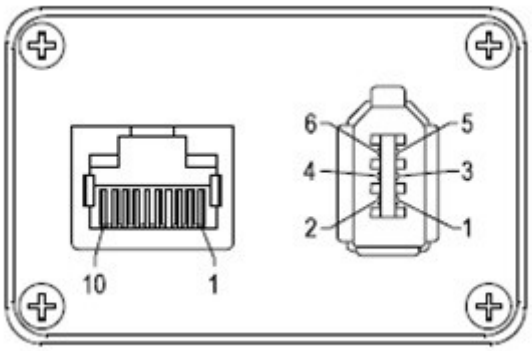
Kamera Basler A631f je vzhledově velice prostá, viz obr. 1. Na zadní straně má 2 typy konektorů. Jedná se o 10-ti pinový konektor RJ-45 (někdy označován jako RJ-50) a o 6-ti pinový IEEE 1394 Socket. Rozvržení jednotlivých pinů je na obr. 2. Konektor IEEE 1394 se používá k napájení kamery a k samotnému přenosu dat. RJ konektor se používá k přístupu k 4 vstupním a výstupním portům. Funkce jednotlivých pinů pro IEEE socket jsou rozepsány v tabulce 2 a pro RJ-45 v tabulce 3.

Já pro propojení kamery s PC používám kabel FireWire a IEEE 1394 řadič, který podporuje napájení kamery. Ne všechny zásuvné moduly jsou toho totiž schopné. Maximální délka propojení mezi kamerou a počítačem nesmí být větší než 4,5 metru.

Testoval jsem i variantu, je-li možné napájet kameru pomocí externího zdroje přes konektor RJ-50 a zároveň s kamerou komunikovat přes FireWire kabel připojený k řadiči, který nepodporuje napájení. Takovéto zapojení se však ukázalo jako nefunkční.



Obr. 1: Kamera Basler A631f [1]



Obr. 2: Číslování pinů pro oba konektory [1]

Tabulka 2: Význam jednotlivých pinů pro FireWire socket

Pin	Označení
1	Napájení (od 8,0 V do 36,0 V)
2	Zem
3	Kroucený pár B -
4	Kroucený pár B +
5	Kroucený pár A -
6	Kroucený pár A +

Tabulka 3: Význam jednotlivých pinů pro RJ-50 socket

Pin	Označení
1	Výstupní Port 3 -
2	Výstupní Port 2 -
3	Výstupní Port 1 -
4	Výstupní Port 0 -
5	Vstupní Port 0+
6	Zem pro všechny vstupy
7	Napájení pro všechny výstupy
8	Vstupní Port 2 +
9	Vstupní Port 1 +
10	Vstupní Port 3 +

1.3 Instalace kamery

1.3.1 Kompatibilita

První krok, který jsem musel udělat, byl zjistit, jestli je kamera kompatibilní s programem MATLAB a jeho součástí Image Acquisition Toolbox. O tomto toolboxu budu pojednávat v další kapitole. Teoretickou kompatibilitu jsem ověřil na webových stránkách výrobce MATLABu, Mathworks. Na následujícím výřezu jedné z webových stránek jsou vypsány podporované typy kamer, viz obr. 3. Výrobce Basler je černě zvýrazněn.

Supported Hardware – Industry Standard – DCAM Compatible FireWire Cameras (IIDC 1394)

Image Acquisition Toolbox supports digital cameras that follow the IIDC 1394-based Digital Camera Specification (DCAM), developed by the 1394 Trade Association. The IIDC 1394-based DCAM specification describes a generic interface for exchanging data with IEEE 1394 (FireWire) digital cameras.

Manufacturers

Many manufacturers provide DCAM cameras that are compatible with Image Acquisition Toolbox, including those in the list below.

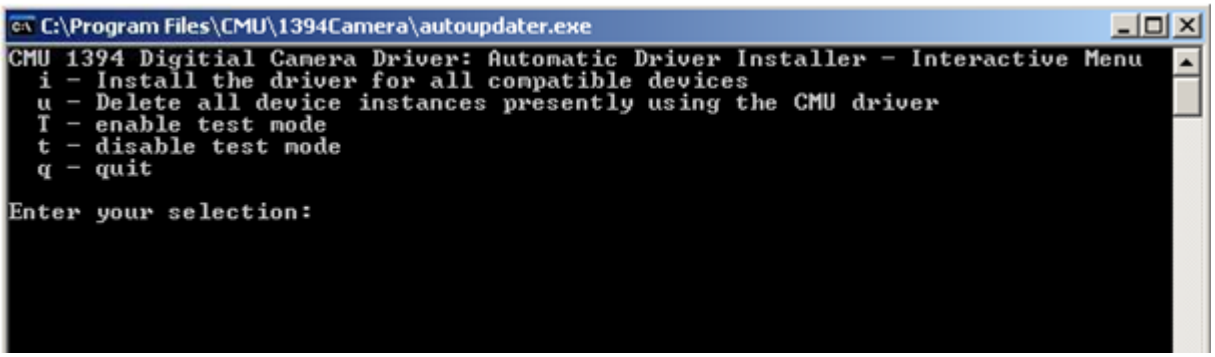
- Allied Vision Technologies
- Basler
- Baumer
- PixeLINK
- Point Grey
- Sony
- Toshiba Teli

Obr. 3: Seznam výrobců podporovaných kamer [2]

1.3.2 Instalace ovladače

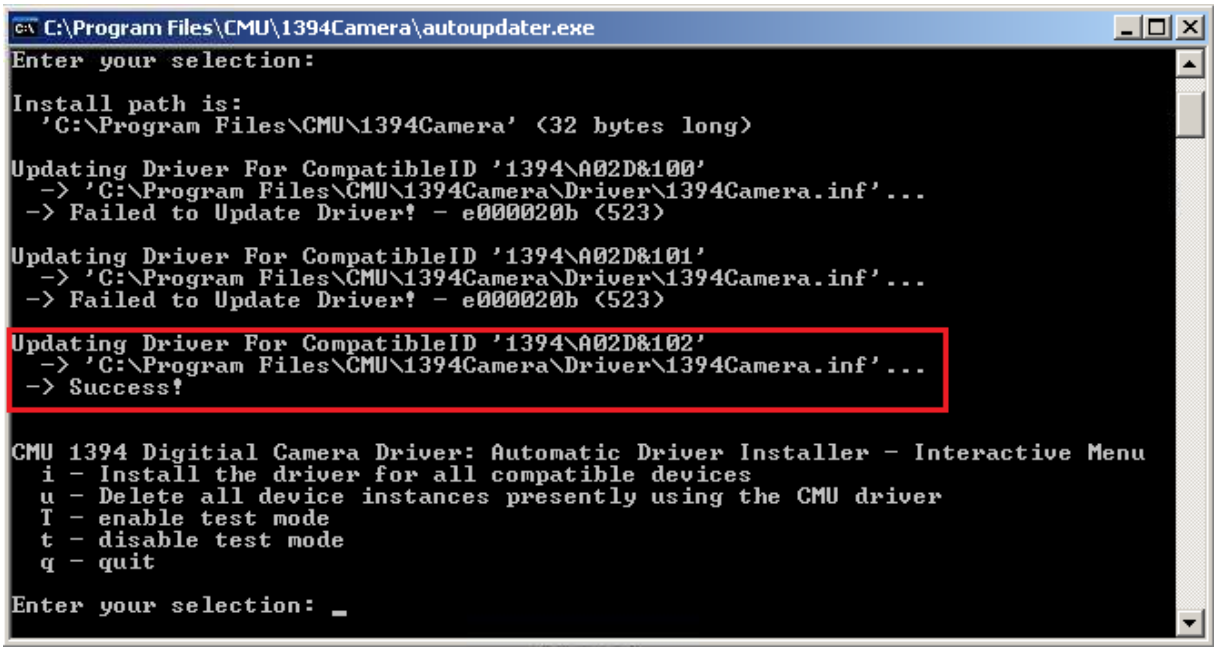
Jakmile jsem se ujistil, že kamera je schopná komunikovat s MATLABem, přistoupil jsem k dalšímu kroku. Tím bylo nainstalovat správný ovladač, který by mi umožnil pracovat s kamerou v prostředí MATLAB a Image Acquisition Toolbox.

Kde najít správný ovladač jsem zjistil opět na internetových stránkách Mathworks [3]. Stáhl jsem a nainstaloval tedy potřebný program 1394Camera Demo. Tento program poskytuje nejen správný ovladač, ale také aplikaci, která slouží k vyhledání a otestování kompatibilních kamer. Ve složce je soubor „autoupdater.exe“, který v pár jednoduchých krocích nainstaloval ovladač kamery. Po spuštění se objevilo okno, které mi dalo na výběr z několika možností, viz obr. 4.



Obr. 4: Úvodní okno pro instalaci ovladače

Zvolil jsem tedy první možnost, protože jsem potřeboval nainstalovat CMU 1394 Digital Camera Driver. Po stisknutí klávesy `i` updater nainstaloval ovladač během několika vteřin. V průběhu instalace jsem pouze musel potvrdit pokračování procesu, protože Windows mne upozornil, že ovladač není digitálně podepsaný a není si jistý, zda bude fungovat. Výrobce ovladače ovšem na toto reaguje ve svém návodu, kde píše, že se toto tvrzení má ignorovat. Jak ukazuje obr. 5, byl driver úspěšné nainstalován.

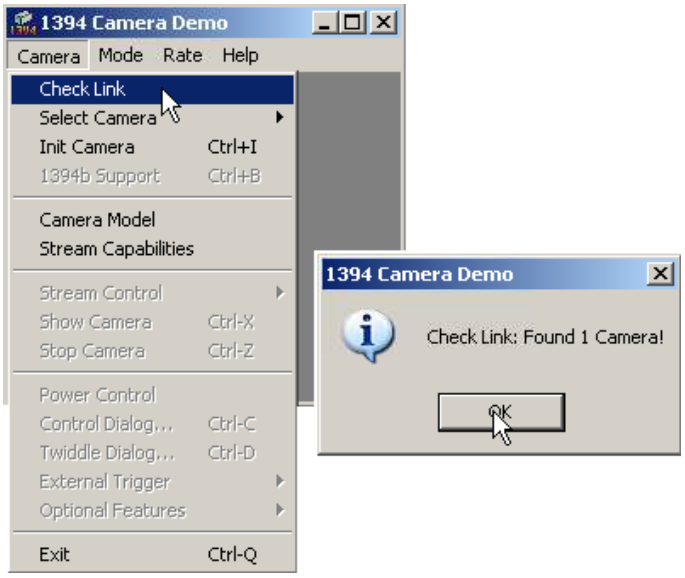


Obr. 5: Zpráva o úspěšném nainstalování ovladače

Instalaci jsem poté opustil stisknutím klávesy `q`.

1.3.3 Otestování kamery

Jak bylo již uvedeno výše, zároveň s ovladačem se nainstalovalo demo aplikace, ve které jsem funkčnost kamery otestoval. Po spuštění bylo nutné rozbalit záložku „Camera“ a poté kliknout na „Check Link“. Program našel 1 kameru, viz obr. 6.



Obr. 6: Menu programu 1394 Camera Demo

Vzhledem k tomu, že program našel pouze jednu kameru, bylo hned toto zařízení vybráno v položce „Select Camera“ a já mohl přejít k otestování. Pak už jen stačilo vybrat „Init Camera“ a po inicializaci „Show Camera“, které již nebylo zašedlé. Odzkoušel jsem všechna podporovaná rozlišení a k nim dostupné frekvence snímkování a vše fungovalo podle očekávání.

2 MATLAB

2.1 Úvod k MATLABu

MATLAB (odvozeno z MATrix LABoratory) je integrované prostředí pro vědeckotechnické výpočty, modelování, návrhy algoritmů, simulace, analýzu a prezentaci dat, paralelní výpočty, měření a zpracování signálů, návrhy řídicích a komunikačních systémů. Je to nástroj jak pro pohodlnou interaktivní práci, tak pro vývoj širokého spektra aplikací.

Výpočetní systém MATLAB se během uplynulých let stal celosvětovým standardem v oblasti technických výpočtů a simulací ve sféře vědy, výzkumu, průmyslu i v oblasti vzdělávání. Svým uživatelům poskytuje nejen mocné grafické a výpočetní nástroje, ale i rozsáhlé specializované knihovny funkcí spolu s výkonným programovacím jazykem čtvrté generace. Knihovny jsou svým rozsahem využitelné prakticky ve všech oblastech lidské činnosti.

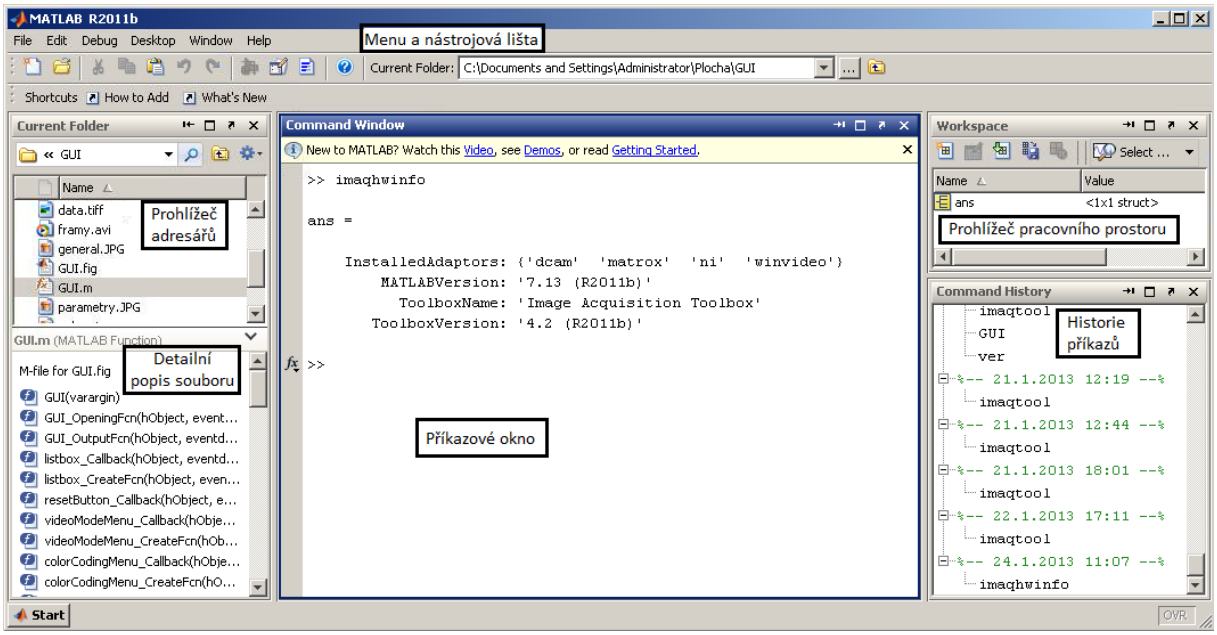
Díky své architektuře je MATLAB určen zejména těm, kteří potřebují řešit početně náročné úlohy a přitom nepotřebují zkoumat matematickou podstatu problémů. Za jeho nejsilnější stránku je považováno mimořádně rychlé výpočetní jádro s optimálními algoritmy, které jsou prověřeny léty provozu na špičkových pracovištích po celém světě. MATLAB byl implementován na všech významných platformách (Windows, Linux, Solaris, Mac).

Vlastností, která nejvíce přispěla k rozšíření tohoto multifunkčního programu do celého světa, je bezpochyby jeho otevřená architektura. MATLAB je úplný programovací jazyk, což znamená, že uživatelé v něm mohou vytvářet funkce detailně přizpůsobené pro jejich aplikace. Tyto funkce se způsobem volání nijak neliší od vestavěných funkcí a jsou uloženy v souborech v čitelné formě. Dokonce většina funkcí s MATLABem dodávaných je takto vytvořena a opravdu vestavěné jsou jen funkce základní. To má dvě velké výhody. Jazyk MATLABu je téměř neomezeně rozšiřitelný a kromě toho se uživatel může při psaní vlastních funkcí poučit z dodaných algoritmů. Navíc jsou takto koncipované funkce snadno přenosné mezi různými platformami, na kterých je MATLAB implementován. Všechny moduly systému doprovází rozsáhlá hypertextová on-line dokumentace, která uživatelům usnadňuje orientaci ve všech funkcích.

MATLAB je koncipován tak, aby kromě pohodlné interaktivní práce umožňoval i programování aplikací. Programovací jazyk obsahuje všechny nezbytné příkazy pro psaní programů, jako jsou podmíněné příkazy, větvicí příkazy, cykly a podobně.

Základním nástrojem výpočetního systému je uživatelské rozhraní MATLAB Desktop. Pracovní nástroje jako prohlížeč adresářů a souborů, prohlížeč pracovního prostoru, okno historie příkazů, interaktivní spouštěč aplikací, editor, debugger, profiler, hypertextová nápověda a příkazové

okno jsou do prostředí plně integrovány, viz obr. 7. Uživatelské rozhraní je konfigurovatelné, takže si uživatel může přizpůsobit rozměry a počet zobrazených nástrojů tak, aby to maximálně vyhovovalo jeho potřebám. Je tak možné vytvořit pracovní plochu, která vyhovuje jak začátečníkům tak pokročilým.



Obr. 7: Prostředí programu MATLAB

Jedná-li se o M-file, obsahuje panel „Detailní popis souboru“ seznam obsažených funkcí. Pokud vybereme z adresáře například *.jpg soubor, objeví se jeho náhled a informace o rozlišení. Do příkazového okna můžeme psát nejen početní operace, ale můžeme pomocí něj spustit všechny součásti MATLABu (Guide, Imaqtool) a také *.fig soubory.

K vytvoření dokonalé grafické podoby uživatelské aplikace pomáhá interaktivní nástroj pro vytváření uživatelských rozhraní, ve kterém lze snadno a přehledně vytvořit a uspořádat ovládací prvky aplikace. Tento nástroj se nazývá GUIDE a podrobněji jsou jeho funkce popsané v kapitole 2.3.

Otevřená architektura MATLABu vedla ke vzniku knihoven funkcí, nazývaných toolboxy, které rozšiřují použití programu v příslušných vědních a technických oborech. Mezi tyto toolboxy patří právě i Image Acquisition Toolbox, který ve své bakalářské práci využívám já a který mi pomáhá při tvorbě a implementaci GUI. Tyto knihovny, navržené v jazyce MATLABu, nabízejí předzpracované specializované funkce, které je možno rozšiřovat a modifikovat. [4]

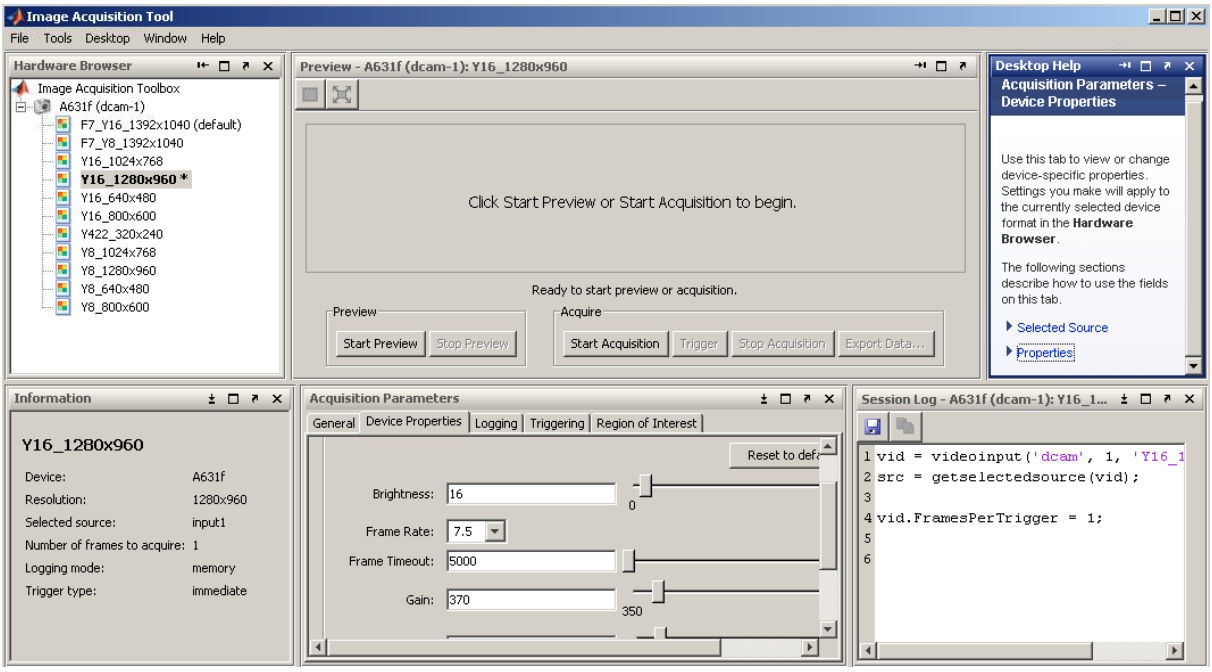
Velice užitečným nástrojem MATLABu je jeho nápověda. V příkazovém okně se spouští pomocí příkazu `doc` a názvu funkce, u které potřebujete zobrazit nápovědu (např. `doc Preview`).

2.2 Image Acquisition Toolbox

Image Acquisition Toolbox (dále už jen Imaqtool) je knihovna funkcí vyvinutá pro počítačové vidění. Umožňuje získat a nahrát samostatné obrazy ale i video ze snímacích zařízení, které jsou s ním kompatibilní, přímo do MATLABu. Tato zařízení umí Imaqtool detekovat automaticky. Po nalezení připojených hardwarovým zařízení lze konfigurovat jejich nastavení. Je zde možné nastavit různé parametry a atributy. Konkrétně u kamery Basler A631f se jedná o tyto nejdůležitější funkce [5][6]:

- rozlišení
- formát videa
- jas
- clona
- zisk
- fps, fpt
- ROI
- okamžitý/manuální trigger

Dále umožňuje zobrazit náhled, který lze přizpůsobovat svým požadavkům za běhu kamery, a v neposlední řadě poskytuje možnost uložit snímek nebo video do paměti nebo přímo na disk. Za pomoci tohoto toolboxu je možno ukládat videa přímo v *.avi formátu. Rozvržení prvků imaqtoolu s připojenou kamerou je vyobrazeno na obr. 8. Tento toolbox se spouští zadáním příkazu `imaqtool` do příkazového okna MATLABu.



Obr. 8: Prostředí Image Acquisition Toolboxu

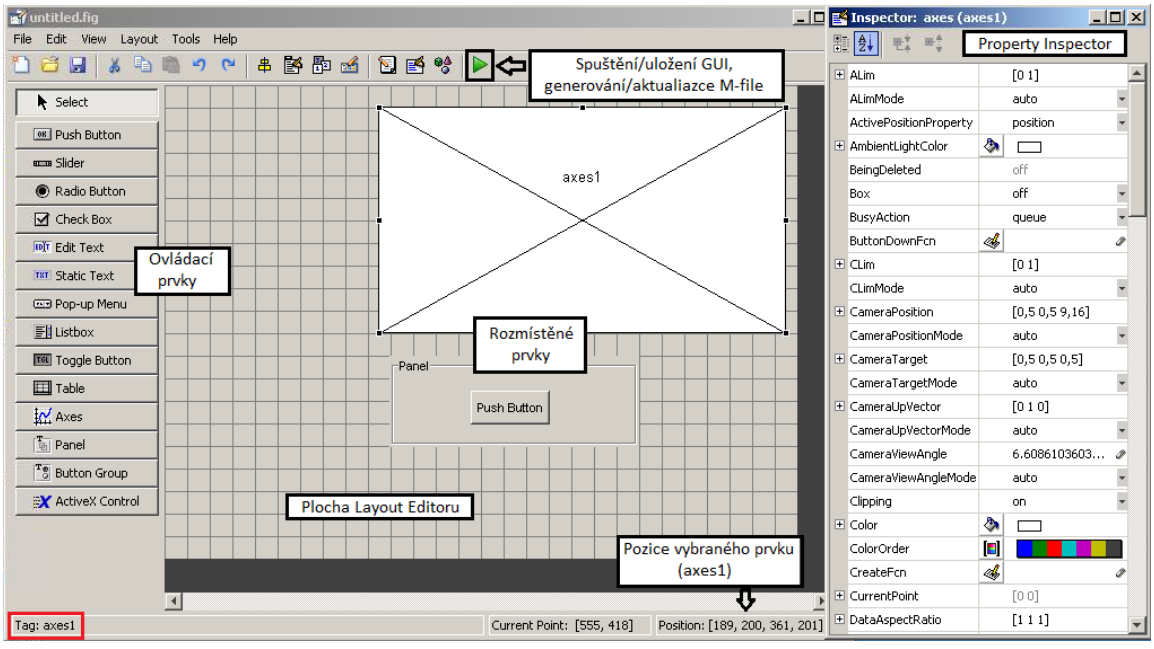
Na levé straně v horní části si můžeme vybrat, jaký formát a rozlišení požadujeme. Ve spodní části jsou základní informace o zařízení. Ve střední části obrazovky je oblast s náhledem a záložky s ovládacími prvky a vpravo je kontextová nápověda a panel „Session Log“.

Panel „Session Log“ je pro mou bakalářskou práci velice užitečný, protože díky němu vidím, jaké příkazy odpovídají jednotlivým akcím, které se v imaqtoolu provádí. Některé tyto pokyny mohu využít pro tvorbu svého grafického uživatelského prostředí. Poupravením části příkazu jej mohu adaptovat pro svou sekvenci instrukcí, která odpovídá například zmáčknutí tlačítka v GUI.

2.3 GUIDE

Tato funkce MATLABu je pro mne stěžejní. Jedná se o grafické uživatelské vývojové prostředí, zjednodušeně grafický editor (Layout Editor), které poskytuje sadu nástrojů pro tvorbu grafických uživatelských rozhraní. Pracovní plocha GUIDE je na obr. 9.

Tyto nástroje usnadňují tvorbu GUI, protože návrh rozhraní tvoříme přetažením jednotlivých komponent (**panely**, **push buttony**, **slidery**, **edit text boxy** aj.) na plochu Layout Editoru, která představuje okno a pozadí budoucí aplikace. Tyto přednastavené komponenty můžeme dále upravovat. Můžeme jim například měnit velikost nebo nastavit inicializační hodnotu. Každá komponenta má totiž své vlastnosti, které jsou shrnuty v podokně „Property Inspector“, viz obr. 9. Zde můžeme modifikovat všechny atributy, které daná komponenta poskytuje. Může to být například barva, velikost písma, umístění, již zmiňovaná velikost či hodnota anebo název neboli Tag (na obrázku vyznačen vyznačen červeně). Tag je pro další programování důležitý, proto je vhodné ho volit výstižně. [7]



Obr. 9: Prostředí GUIDE a Property Inspector

Při uložení návrhu GUI (soubor bude mít příponu *.fig) se automaticky vygeneruje tzv. M-file (soubor s příponou *.m), který mimo jiné obsahuje tzv. Callbacky a kde se programuje funkčnost GUI. Každá komponenta z GUI má v M-souboru svůj Callback (zpětnou vazbu) nazvaný právě podle Tagu (například startButton_callback). Sekvence příkazů, kterou naprogramujeme do příslušného Callbacku, poté zajistí, aby se událo přesně to, co se má stát. Například aby se po stlačení tlačítka Start spustil náhled kamery.

Editor GUIDE se spouští zadáním příkazu `guide` do příkazového okna MATLABu.

2.3.1 Obecná pravidla pro tvorbu GUI

Při vytváření vzhledu aplikace musíme brát v úvahu několik principů a pravidel, které se postupem času dostaly do podvědomí a staly se jakýmsi standardem. Tvůrce GUI se musí nejdříve zamyslet nad pár základními věcmi:

- cílová skupina – pro koho je aplikace určena (pro děti, úředníky, zkušené uživatele, vědce)
- druh aplikace – k čemu se bude aplikace používat (hraní, psaní, počítání, vědecké účely)

Dalším krokem je samotný návrh GUI a rozvržení komponent. Před vlastní tvorbou v grafickém editoru je velice vhodné si nejprve nakreslit vzhled budoucího GUI na papír. Rozvrhneme, jak budou jednotlivé prvky poskládány na ploše a jakou by tyto komponenty měly mít funkčnost.

Má cílová skupina jsou zkušení uživatelé. Kamera je připojena k mikroskopu a používá se ke zkoumání a analýze snímků (např. pro zkoumání struktury optického vlákna). I když jednoduchost aplikace může značit opak, jedná se o GUI pro účely vědeckého zkoumání.

Posledním krokem je samotná tvorba uživatelského rozhraní, u kterého bychom měli dodržovat většinu následujících zásad [8][9]:

- jednoduchost a elegance návrhu,
- konzistence funkcionality – stejné komponenty by měly mít obdobnou funkci,
- rozumné používání barev, kontrastů, rozměrů, efektů a animací,
- jednotný styl – stejná velikost písma, barva, rozměry stejných prvků,
- použití zažitých značek – například pro pohyb použít symbol šipek,
- vyhnout se zmatení – při složitějších krocích vést uživatele, aby se neztratil,
- vyladit chování komponent při zvětšování/zmenšovaní okna aplikace,
- myslet dopředu na případné rozšiřování GUI.

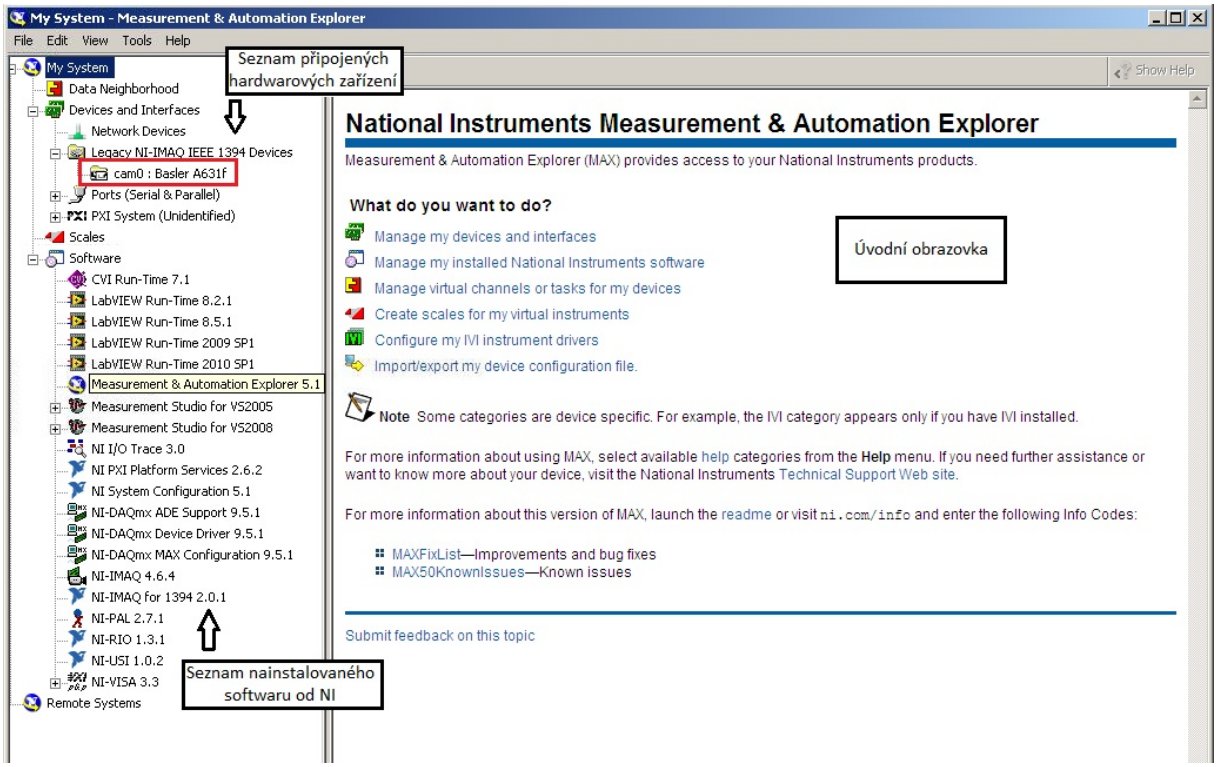
3 Measurement & Automation Explorer

Measurement and Automation Explorer (MAX), který rovněž využívám ve své bakalářské práci, je produktem firmy National Instruments (NI). Je softwarovou součástí všech ovladačů dodávaných firmou NI. Pomocí tohoto programu můžeme kontrolovat připojená a nainstalovaná zařízení a testovat jejich správnou funkčnost. Také je zde přehled nainstalovaného softwaru přímo od National Instruments, který můžeme odsud spouštět a aktualizovat.

MAX je grafické uživatelské rozhraní sloužící také ke konfiguraci virtuálních přístrojů. Konfigurace probíhá přes neustále vyvíjený a rozšiřovaný ovladač pro zařízení DAQ firmy National Instruments nazývající se NI-DAQmx. MAX přečte informace v ovladači zařízení v registrech operačního systému Windows a přiřadí zařízení logické jméno pro NI-DAQmx. [10]

3.1 Popis prostředí

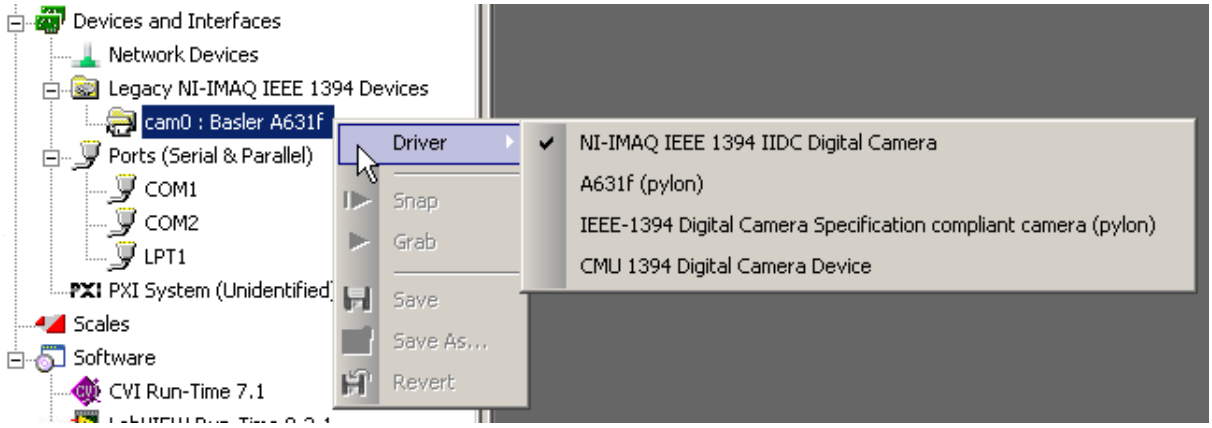
Na obr. 10 je popsáno prostředí systému MAX po jeho spuštění. Kamera Basler je zde červeně vyznačena.



Obr. 10: Prostředí Measurement and Automation Exploreru

Po kliknutí na vybraný hardware se místo úvodní obrazovky objeví jeho pracovní prostředí. U naší kamery se jedná o snímací oblast a oblast s nastavením a atributy. Vybereme-li software, objeví se tabulka s nainstalovanou verzí, krátkým popisem a cestou k adresáři, kde je software nainstalován.

Za velikou výhodu považuji to, že po nainstalování 2 a více různých ovladačů pro hardwarové zařízení si je MAX dokáže zapamatovat a umožní nám mezi těmito ovladači přepínat, jak je vidět na obr. 11. Konkrétně pro mou kameru potřebuji ovladač pro NI, abych mohl zkoumat předlohu pro své GUI a jeho funkčnost přímo v prostředí MAX, a CMU 1394 DCAM driver, který umožňuje kameře komunikovat s MATLABem.



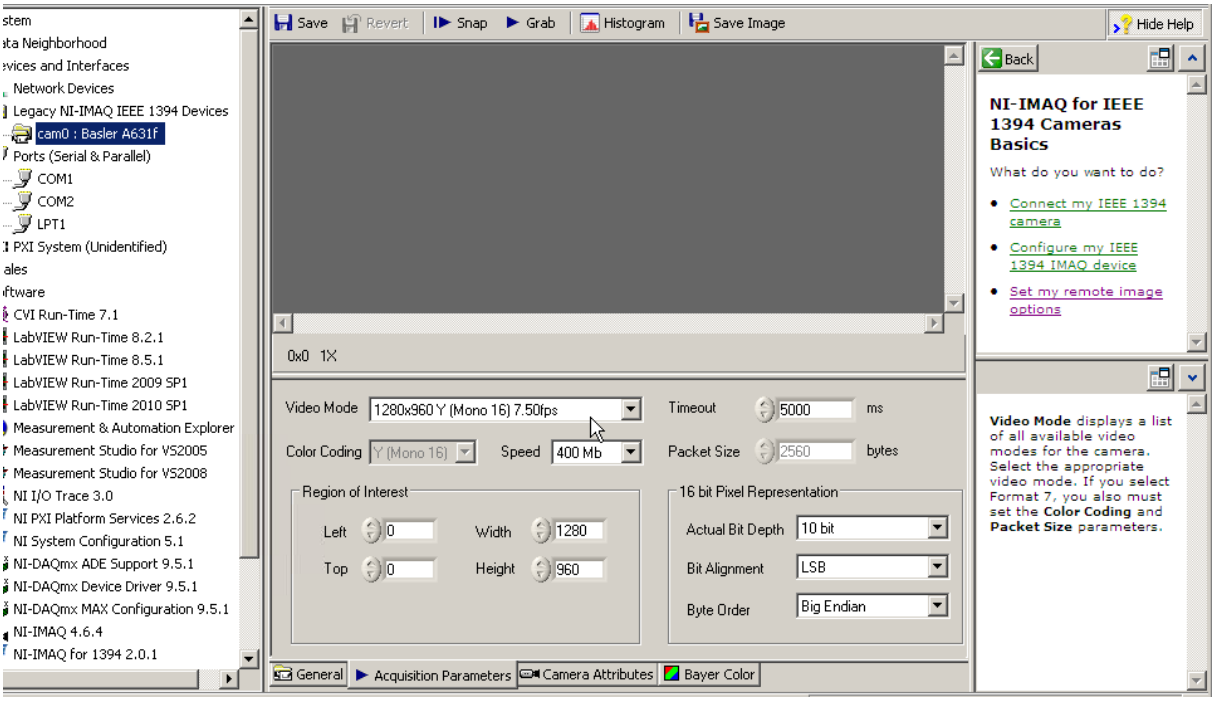
Obr. 11: Rozklikávací menu pro přepínání mezi ovladači

3.2 Vzhled GUI

Na obr. 12 je GUI tak, jak je vyobrazeno v systému MAX. V podstatě je rozděleno na dvě poloviny. V horní polovině je snímací oblast a ovládací pruh, který umožňuje spouštět kameru, ukládat snímky anebo zobrazovat histogram. V dolní polovině je nastavovací oblast. Jsou zde vyobrazeny všechny atributy a nastavení, která kamera poskytuje a která se mohou upravit. Pravý pruh mimo kamerové GUI slouží jako obecná nápověda a jako informační okno k jednotlivým atributům.

GUI, které vytvářím já, bude mít ve výsledku jiný vzhled. Hlavně z toho důvodu, že GUIDE v MATLABu zatím nepodporuje ovládací prvek záložky v takové šíři funkcionality, v jaké jej potřebuji. Teprve se o něm uvažuje do budoucích verzí. Já pracuji s verzí 7.13.0.564 (R2011b). Musel jsem tedy všechny záložky rozdělit do jednotlivých oddělených oblastí.

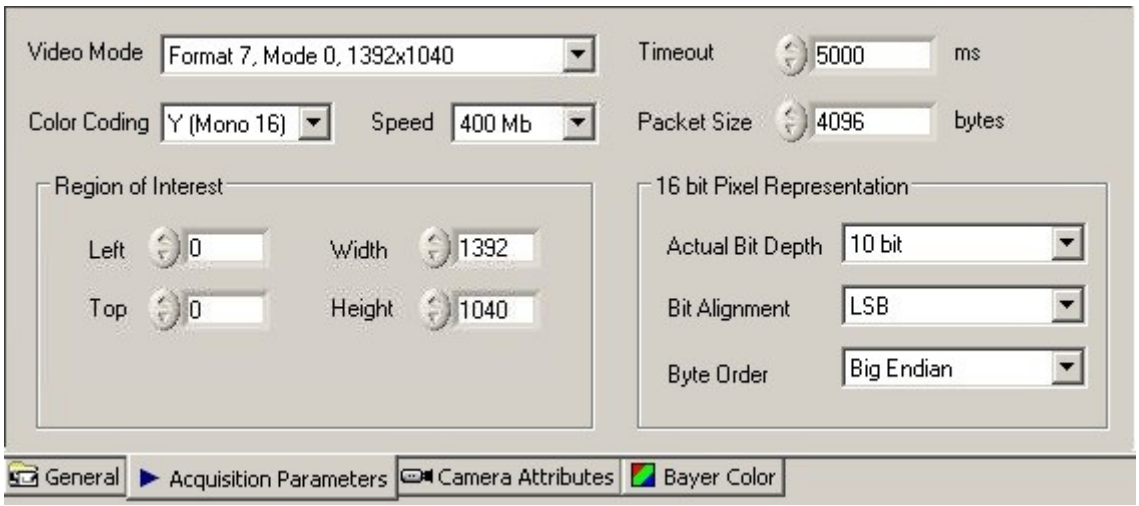
Záložka Bayer Color bude úplně vypuštěna, protože Basler A631f je monochromatický model.



Obr. 12: GUI pro kameru v Measurement and Automation Exploreru

3.2.1 Snímací parametry

Na obr. 13 uvedeném níže je záložka „Acquisition parameters“, na které se nastavují základní vlastnosti kamery.



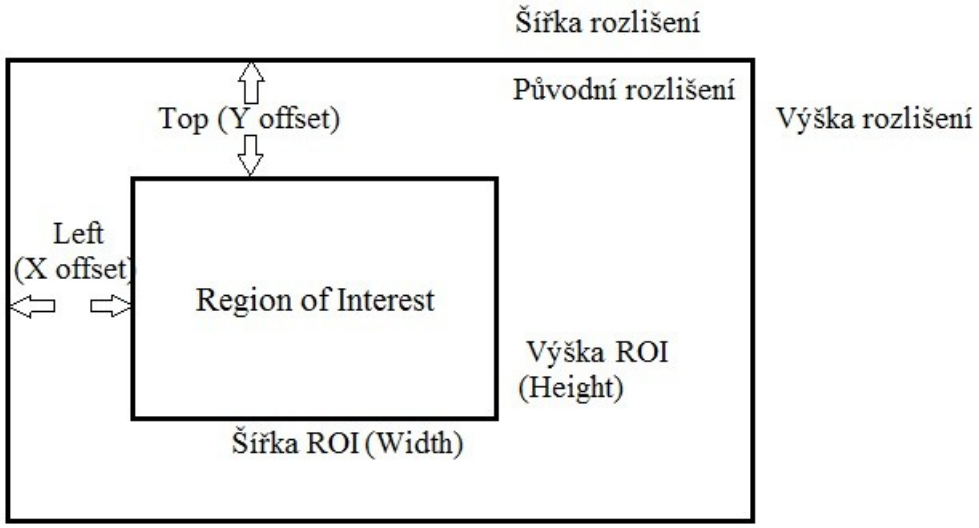
Obr. 13: Záložka Acquisition Parameters v Measurement and Automation Exploreru

Je zde rozbalovací menu „Video mode“, ve kterém si uživatel vybírá rozlišení, video formát a snímkovací frekvenci. Tyto možnosti jsou v menu zkombinované tak, abychom je mohli plně využít v šíři, jakou kamera poskytuje. Rozbalovací menu „Color coding“ je needitovatelné a zobrazuje vybraný video formát, tedy Mono 16, Mono 8 nebo YUV 4:2:2. Položka „Speed“ udává, jakou

rychlostí se budou data přenášet. Dostupné možnosti jsou 100, 200 nebo 400 Mb. Položka „Packet Size“ je dostupná pouze u rozlišení 1392 x 1040 obrazových bodů. Pomocí ní se pro toto rozlišení nastavuje rychlost snímkování, viz vzorec 1. Prvek „Timeout“ specifikuje čas v milisekundách, který je vyhrazen pro uložení snímku do paměti kamery od chvíle jeho pořízení. Následující vzorec popisuje, jak se vypočítá výsledná snímkovací frekvence na základě hodnoty zadané v poli „Packet Size“, která ve vzorci odpovídá proměnné „Packets per Frame“. Jednotkou této veličiny je byte. Hodnota 125 mikrosekund je konstanta udávaná výrobcem. Maximální možná vstupní hodnota je 1024 B, která odpovídá 7,8 fps. Naopak minimální možná vstupní hodnota je u rozlišení s formátem Y8 Mono 427 B, což odpovídá 18,7 fps, a u rozlišení s formátem Y16 Mono 708 B, která udává maximální snímkovací frekvenci 11,3 fps.

$$Frames\ per\ Second\ = \frac{1}{Packets\ per\ Frame\ x\ 125\ \mu s} \qquad (1)$$

Panel „Region of Interest“ (ROI) nám umožňuje nastavit, jakou část snímku chceme zobrazit. Díky němu můžeme vytvořit jakýsi výřez z celé plochy, který nás zajímá více než okolí. Položka „Left“, neboli také X offset, udává, o kolik obrazových bodů z levé strany bude snímaná plocha zmenšena oproti původnímu nastavení. Hodnota „Top“ (Y offset) funguje stejně, ale platí pro odsazení od horní části. Hodnoty „Width“ a „Height“ s těmito offsety souvisí, a proto se podle nich modifikují, aby se zachovala logika funkčnosti. To v praxi znamená to, že součet Left a Width nemůže překročit hodnotu maximální šířky rozlišení. Stejná zákonitost platí pro Top a Height. Na obr. 14 je znázorněna logika Region of Interest.

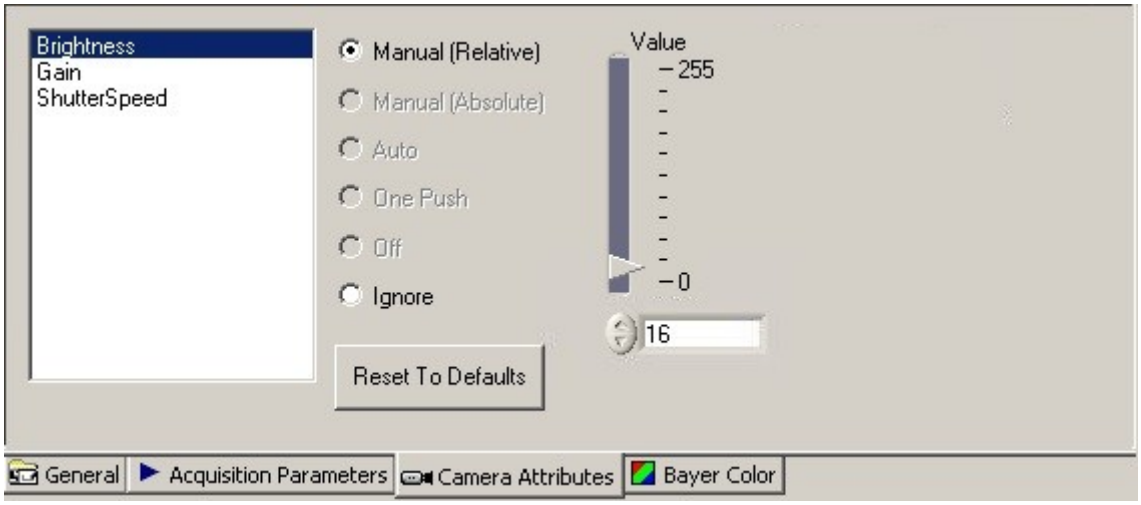


Obr. 14: Princip nastavení Region of Interest

Poslední položkou na této záložce je „16 bit Pixel Represenatiton“. Tento panel je aktivní pouze tehdy, když jsme vybrali ve „Video Mode“ možnost s formátem Mono 16. Jedná se tedy o 16-ti bitové nastavení. „Actual Bit Depth“ definuje počet bitů na 1 pixel obrázku. Na výběr jsou hodnoty 10, 12, 14 a 16 bitů. Dále je zde „Bit Alignment“, kterým můžeme nastavit pořadí bitů, v jakém se budou ukládat (LSB, MSB). Poslední položkou je „Byte Order“, který také souvisí s pořadím ukládání dat. Je možno vybrat Big Endian nebo Little Endian. Big Endian znamená, že na nejnižší adresu se ukládá nejvýznamnější bit (MSB). Little Endian značí, že nejnižší adresa obsahuje nejméně významný bit (LSB).

3.2.2 Vlastnosti kamery

Na této záložce se nachází seznam dostupných atributů kamery. Počet těchto vlastností se liší v závislosti na typu a výrobci kamery. V našem případě kamera Basler A631f nám dává k dispozici upravovat brightness (jas), gain (zisk) a shutter speed (clona). K manuálnímu nastavení hodnot je třeba vybrat příslušný **radio button** (Manual(Relative)) a poté na **slideru** vybrat požadovanou hodnotu. Číslo je také možné vepsat přímo do nabízeného **edit text boxu**. Při vložení desetinného čísla se toto číslo zaokrouhlí na celou část. Nakonec je zde ještě **push button** pro resetování, které všechny dostupné atributy nastaví na defaultní hodnotu. Koncepce této záložky je na obr. 15.



Obr. 15: Záložka Camera Attributes v Measurement and Automation Exploreru

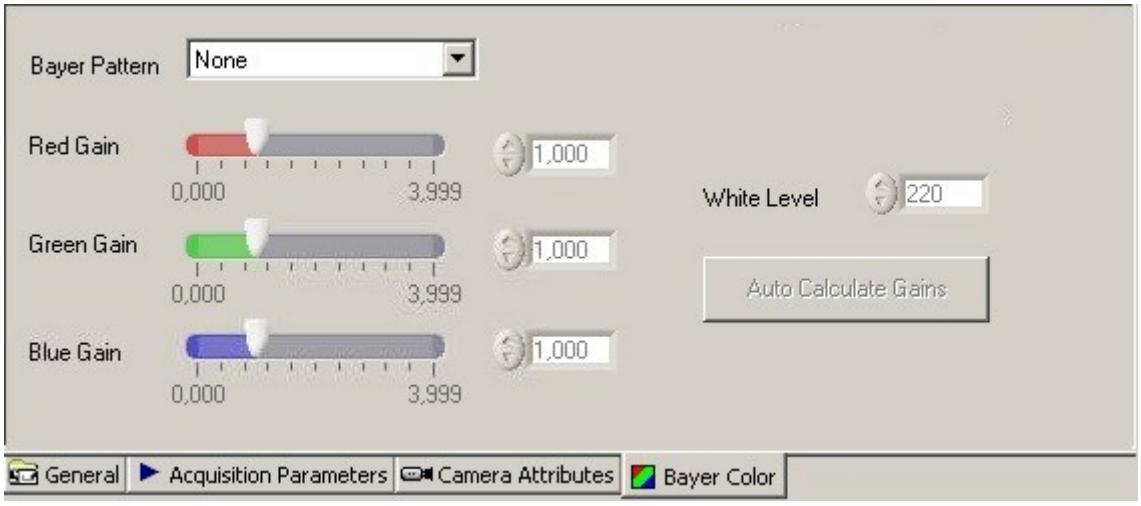
3.2.3 Bayer Color

Tato záložka slouží k nastavení tzv. Bayerova filtru. Jedná se o filtr, který se používá k filtraci světla dopadajícího na snímací čip. Každý pixel je pokryt mikrovrstvou, která propouští pouze jednu barvu. Jedná se o barvy červenou, zelenou a modrou (RGB model). V každém bloku čtyř pixelů (2x2 pixely) jsou 2 pixely, které reagují pouze na zelenou barvu, 1 pixel, který reaguje pouze

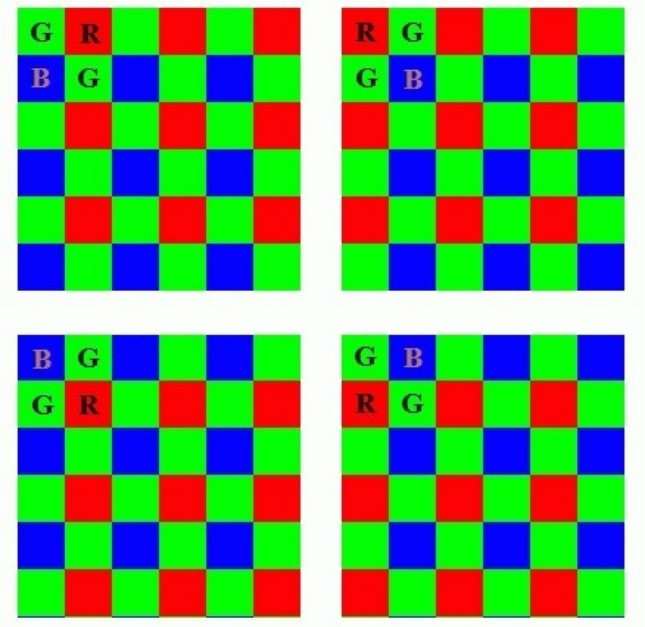
na červenou, a 1 reagující jen na modrou barvu. Tato kombinace byla nastavena tak, aby imitovala citlivost lidského oka vůči barvám, protože oko je nejcitlivější právě na zelenou barvu.

Rozvržení této záložky je na obr. 16.

Rozbalovací menu „Bayer Pattern“ slouží k výběru vzoru filtru. Existují 4 různé varianty filtru, viz obr. 17. Po vybrání jakékoliv varianty se zpřístupní všechny **slidery** a **edit text boxy**, pomocí kterých se tyto vzory mohou dále upravovat. **Push button** „Auto Calculate Gains“ slouží k automatickému výpočtu hodnot jednotlivých barevných složek v závislosti na hodnotě bílé složky.



Obr. 16: Záložka Bayer Color v Measurement and Automation Exploreru

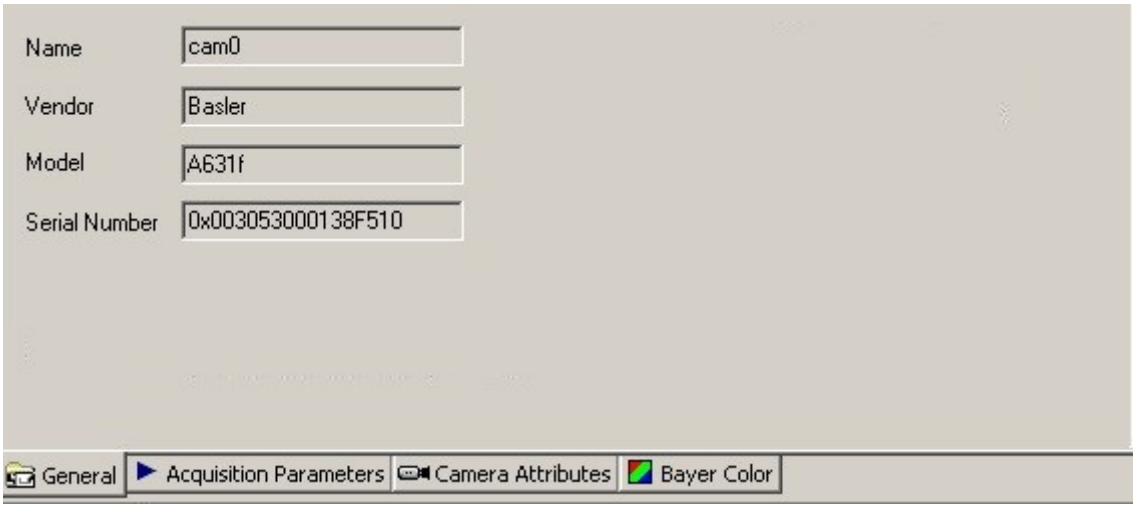


Obr. 17: Možné varianty Bayerova filtru

Tento druh filtru je ale dostupný pouze u barevných kamer, proto u naší monochromatické kamery nemá žádné využití.

3.2.4 Informace o kameře

V poslední záložce „General“ jsou pouze základní informace o kameře, viz obr. 18. Jedná se o výrobce, model, jméno a sériové číslo kamery.



Obr. 18: Záložka General v Measurement and Automation Exploreru

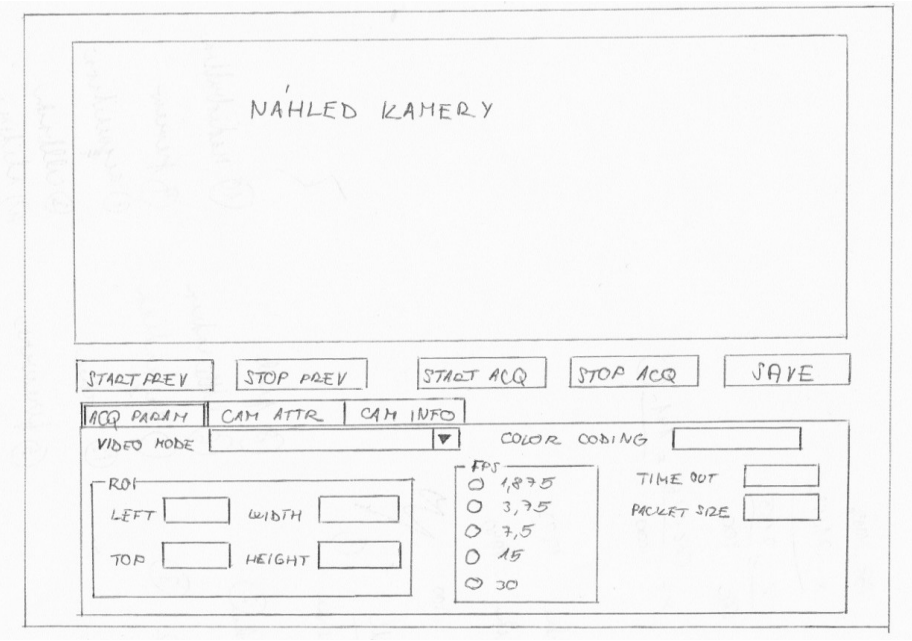
4 GUI pro kameru Basler A631f

4.1 Návrh a vzhled GUI

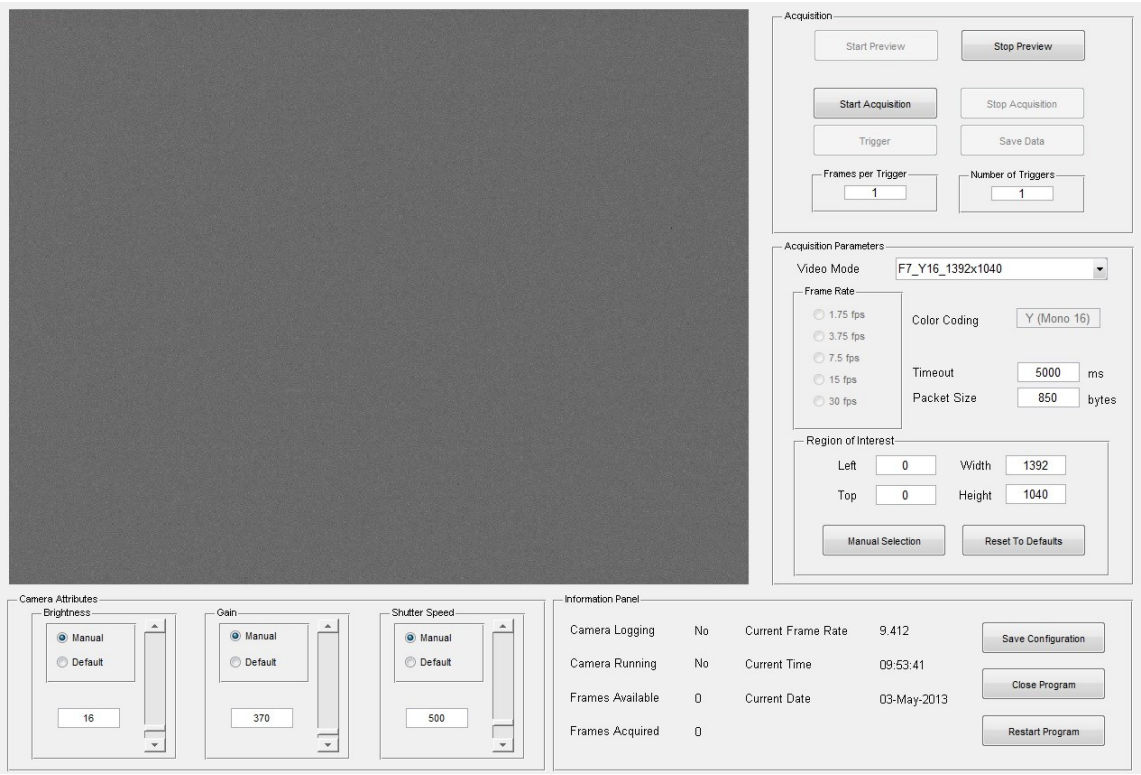
Vzhled mnou vytvořeného uživatelského prostředí se od původního návrhu podstatně liší. Jak je možné vidět na obr. 19, první nákres se velmi podobal GUI z programu MAX. Jak jsem se již zmiňoval v kapitole 3.2, nemohl jsem použít systém záložek, a proto jsem musel všechny záložky rozdělit do jednotlivých panelů a přizpůsobit velikosti komponent velikosti obrazovky. Vzhledem k tomu, že různé monitory jsou různě velké, musel jsem GUI upravit tak, aby bylo univerzální. Ve verzi MATLABu, kterou používám, nejsou pro výslednou aplikaci automaticky dostupné posuvníky, je-li okno aplikace větší, než dovoluje rozlišení. Aplikace je tedy vytvořena tak, aby se komponenty zmenšovaly nebo zvětšovaly podle toho, jak si uživatel velikost okna nastaví.

Konečný vzhled rozhraní je tedy složen z náhledu a čtyř panelů, které svou funkčností odpovídají prostředí MAX. Rozložení komponent je vidět na obr. 20. Celkový koncept je navržen způsobem, který by na uživatele měl působit intuitivně a jednoduše. Oblast s náhledem je nejdůležitější, proto je na ni kladen největší důraz. Dále by uživatel měl postupovat od tlačítek směrem dolů, protože komponenty jsou v tomto směru řazeny podle důležitosti pro správné fungování programu.

Všechny nápisy ve výsledné aplikaci jsou v anglickém jazyce, protože mají odpovídat zobrazení v systému MAX. Veškeré GUI komponenty a prvky, které popisuji v této bakalářské práci, nazývám rovněž anglickými ekvivalenty, aby odpovídaly pojmenováním prvků v prostředí GUIDE.



Obr. 19: Předběžný návrh GUI pro kameru Basler A631f



Obr. 20: Konečný vzhled GUI pro ovládání kamery Basler A631f se spuštěným náhledem

4.2 Manuál pro GUI

Jak jsem již psal výše, ovládání aplikace by mělo být velmi intuitivní. Nejdůležitější komponentou je rozbalovací menu „Video Mode“. Od ní se odvíjí veškerá ostatní funkcionalita GUI. Řetězec znaků reprezentuje jednotlivé vlastnosti. F7 znamená, že uživatel nemá možnost nastavit rychlost snímkování pomocí **radio buttonů**, ale bude mít zpřístupněnou položku „Packet Size“. Jedná se o tzv. Formát 7. Tato funkce je pouze u největšího rozlišení. Další skupinou znaků je Y16, Y8 nebo Y422. Tyto znaky odpovídají textu, který se zobrazí v prvku „Color Coding“ a reprezentují výstupní video formát. Poslední částí jsou vždy dvě čísla, která představují rozlišení výsledného snímku. Podle toho, jakou položku vybereme, se aktivuje panel „Frame Rate“, kde si zvolíme jednu z dostupných snímkovacích frekvencí. Předvolena je vždy největší možná.

Panel „Region of Interest“ slouží k zobrazení určitého výřezu plochy. Číselné hodnoty „Width“ a „Height“ odpovídají rozlišení, které si vybereme. Defaultně je vše nastaveno, aby došlo k zobrazení celého snímku. Funkcionalita ROI je popsána v kapitole 3.2.1. **Push button** „Reset To Defaults“ uvede všechny čtyři číselné hodnoty do základního stavu daného výrobcem. **Toggle button** „Manual Selection“ slouží k tomu, aby uživatel mohl manuálně vybrat oblast, kterou chce zobrazit, na právě běžícím náhledu. Tato funkce je přístupná pouze tehdy, je-li spuštěn náhled. Po jeho prvním stisknutí má uživatel možnost vybrat ROI, které chce. Po jeho druhém stlačení se tyto hodnoty načtou, vypíší do příslušných **edit text boxů** a náhled se podle nich upraví.

Dalším krokem je nastavení dostupných vlastností kamery v panelu „Camera Atrributes“. Každý atribut má svůj panel. Je-li vybrán **radio button** „Manual“, je zpřístupněný **slider** i **edit text box** a můžeme nastavit požadovanou hodnotu. Jakmile bude vybrán **radio button** „Default“, nastaví se defaultní hodnota, kterou udává výrobce.

Všechny tyto atributy mají nastavené maximální a minimální hodnoty, které kamera povoluje. U jasu (brightness) je dostupný interval 0-255. Defaultní hodnota je 16. Pokud se jedná o zisk (gain), musí být hodnota mezi 350 a 1023 a nastavená hodnota při startu je 370. Pro clonu (shutter speed) je možné nastavit hodnotu v rozmezí 1-4095. Defaultní hodnota je 500.

Tyto atributy také částečné souvisí s formátem, který je vybrán ve „Video Mode“. Je-li vybrán formát, který obsahuje Y16, ale nejedná se o formát F7, je maximální hodnota pro zisk 511. Naopak je-li vybráno Y8 nebo Y422, je maximální hodnota již zmíněných 1023. Pro ostatní atributy se intervaly nemění.

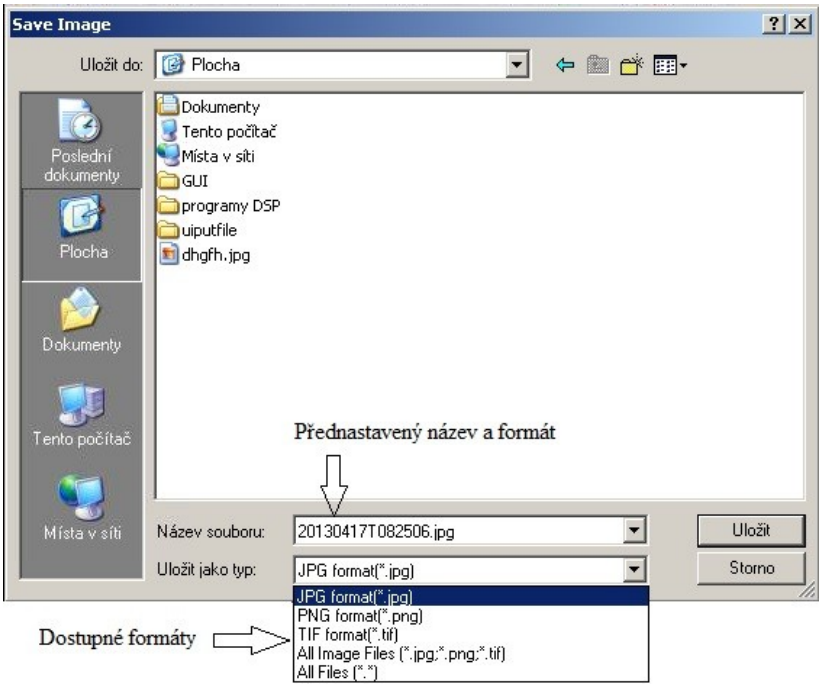
Poslední hodnoty, které jsme ještě nenastavili, jsou „Frames per Trigger“ a „Number of Triggers“. Frames per Trigger je počet framů (snímků), které se vyfotí při jednom stisknutí **push buttonu** „Trigger“. Number of Triggers je počet triggerů, které můžeme během snímání využít. Jedná se tedy o číslo, kolikrát zmáčkneme **push button** „Trigger“ a tím pořídíme snímek. Množství času mezi dvěma nebo více triggery není pevně určen. Výsledný počet framů je dán vzájemným vynásobením hodnot z obou **edit text boxů**.

Posledním panelem, který ještě nebyl popsán je panel „Information panel“. Obsahuje tři **push buttony** a několik informačních položek. Tyto položky poskytují uživateli informace, v jakém stavu se kamera nachází. Je-li kamera připravena ke snímání, bude u „Camera Running“ text „Yes“. Má-li kamera v paměti uložena nějaká data, bude u „Camera Logging“ text „Yes“. V opačných případech bude u těchto položek „No“. „Frames Acquired“ a „Frames Available“ informují uživatele, kolik snímků bylo pořízeno a kolik jich je uloženo v paměti kamery. To znamená, že „Frames Acquired“ se nebude měnit a bude stále zobrazovat počet pořízených snímků v rámci jednoho snímání. Naopak „Frames Available“ se bude snižovat podle toho, jak bude uživatel ukládat jednotlivé snímky. Tyto čtyři položky se po opětovném stisknutí „Start Acquisition“ vyresetují a znovu budou zobrazovat informace k aktuálnímu snímání. Položky „Current Date“ a „Current Time“ zobrazují aktuální datum a čas. Posledním prvkem je „Current Frame Rate“. Ten zobrazuje aktuální snímkovací frekvenci kamery. Tento údaj je užitečný hlavně ve chvíli, kdy pracujeme s nejvyšším rozlišením a snímkovací frekvenci nastavujeme pomocí **edit text boxu** „Packet Size“. **Push button** „Save Configuration“ vytvoří textový soubor v adresáři, ve kterém je uloženo GUI, a uloží do něj všechny hodnoty a atributy, které uživatel nastavil. Tato funkce je zde proto, aby si uživatel uložil nastavení, které mu

nejlépe vyhovuje a které by chtěl někdy v budoucnu použít. **Push buttony** „Close Program“ a „Restart Program“ slouží k ukončení aplikace nebo k jejímu restartování.

Máme-li vybráno a nastaveno vše, co potřebujeme, můžeme přejít k náhledu či snímání. I kdyby uživatel zapomněl některé hodnoty nastavit, je GUI definováno tak, aby byly použity defaultní hodnoty, a proto dojde ke spuštění náhledu. Náhled se spouští **push buttonem** „Start Preview“ a ukončuje se pomocí „Stop Preview“. Náhled slouží pouze k tomu, abychom viděli, jak bude vypadat výsledný snímek. Nelze tento snímek vyfotit či uložit. Během spuštěného náhledu lze měnit jednotlivé atributy kamery a tyto změny se okamžitě projeví na náhledu. Hodnoty, které lze měnit jsou přístupné, ostatní jsou v danou chvíli zakázané. Po stisknutí „Stop Preview“ budou všechny komponenty opět zpřístpněny.

Chceme-li zahájit snímání, je třeba kliknout na „Start Acquisition“. Objeví se náhled a zároveň se zpřístupní **push buttony** „Trigger“ a „Stop Acquisition“. **Push button** „Stop Acquisition“ se používá tehdy, chceme-li ještě něco přenastavit nebo například změnit rozlišení a v případě, jsou-li dokončeny všechny triggery. Jeho stisknutím totiž opět zastavíme náhled a kameru. Trigger je tzv. spouštěč, který po stisknutí vyfotí daný snímek. Pokud jsme pořídili alespoň jeden snímek, zpřístupní se po kliknutí na „Stop Acquisition“ **push button** „Save Data“. Po stisknutí uložení se otevře klasický průzkumník pro vybrání cílové složky a jména souboru, viz obr. 21. Přednastavený název souboru obsahuje čas a datum, kdy byl snímek pořízen. Můžeme si vybrat z několika dostupných formátů pro uložení obrázku. Jakmile dokončíme ukládání, dostaneme se opět do první fáze, ve které můžeme nastavit jiné hodnoty atributů, změnit rozlišení atd.



Obr. 21: Okno průzkumníku pro ukládání dat z paměti kamery

4.3 Detailní popis funkčnosti aplikace

4.3.1 Vytvoření aplikace a inicializace

Jak jsem uvedl v kapitole 2.3, po uložení návrhu GUI v GUIDE, se vygeneruje M-file. Jsou v něm obsaženy všechny callbacky všech komponent, které jsme použili, ale také dvě funkce navíc. První funkce obsahuje kód, do kterého se nevpisuje a který slouží k otevření vlastního figure, které jsme si vytvořili v GUIDE. Druhá funkce je `function GUI_OpeningFcn`, která slouží jako inicializační. Sekvence příkazů vepsaná do této funkce se vykoná v okamžiku, kdy se aplikace objeví na obrazovce. V mém případě se jedná o instrukce, které načtou do příslušných proměnných hodnoty potřebné ke spuštění náhledu nebo snímání.

Pro ukázku jsem vybral nastavení minima, maxima a inicializační hodnoty **slideru** pro zisk (`handles.gainSlider`). Pro tento atribut je nastavena hodnota 370.

```
set(handles.gainSlider, 'Min', 350);
set(handles.gainSlider, 'Max', 1023);
set(handles.gainSlider, 'Value', 370);
set(handles.gainEdit, 'String', '370');
```

`Handles.axes1` je objekt, ve kterém se zobrazuje náhled kamery. Aby uživatele nerušila bílá obrazovka při spuštění aplikace, vypnul jsem jeho viditelnost.

```
set(handles.axes1, 'Visible', 'off');
```

Abych uživatele vedl k správnému používání aplikace, pro jednotlivé **push buttony** jsem nastavil, zda se mohou v danou chvíli stisknout či nikoliv.

```
set(handles.startPreviewButton, 'Enable', 'on');
set(handles.stopPreviewButton, 'Enable', 'off');
set(handles.startAcquisitionButton, 'Enable', 'on');
set(handles.stopAcquisitionButton, 'Enable', 'off');
set(handles.triggerButton, 'Enable', 'off');
set(handles.saveDataButton, 'Enable', 'off');
```

Na konci každé funkce, která předává jakákoli data jiné funkci, musí být následující řádek. Právě on zajišťuje, aby ostatní příkazy mohly používat nastavené hodnoty uživatelem. Nemusíme ovšem předávat vždy všechny proměnné, globalizovány budou pouze ty, které mají `handles.` připsáno před svým názvem (např. `handles.axes1`).

```
guidata(hObject, handles);
```

4.3.2 Snímací parametry

Jedna z nejdůležitějších funkcí v aplikaci je `function videoModeMenu_Callback`. Je to callback, který nastavuje rozlišení videa, jeho barevný (respektive monochromatický) formát,

velikost paketu (packet size), snímkovací frekvenci (fps) a maximální hodnotu zisku. Pro každé dostupné rozlišení je podobná sekvence příkazů, která se liší v koncových hodnotách. Kamera nabízí 11 různých rozlišení v kombinaci s barevným formátem. Pro představu jsem vybral maximální rozlišení a případ, kdy je zpřístupněn panel „Frame Rate“.

Rozlišení 1392x1040 pixelů, Mono 16:

```
% Proměnná, která hlídá, zda již byl vytvořen videoobjekt či ne
      handles.kamera = 0;
% Zapíšu šířku rozlišení do pole widthEdit
      set(handles.widthEdit, 'String', '1392');
% Zapíšu výšku rozlišení do pole heightEdit
      set(handles.heightEdit, 'String', '1040');
% Do proměnné šířka uložím hodnotu widthEdit převedenou na číslo
      handles.sirka = str2num(get(handles.widthEdit, 'String'));
% Do proměnné výška uložím hodnotu heightEdit převedenou na číslo
      handles.vyska = str2num(get(handles.heightEdit, 'String'));
% Proměnné leftEdit a topEdit nastavím na 0
      set(handles.leftEdit, 'String', '0'); % Nastavím leftEdit
      set(handles.topEdit, 'String', '0'); % Nastavím topEdit
% Nastavím všechny radiobuttony v panelu Frame Rate na needitovatelné,
protože nejvyšší rozlišení se jimi neřídí
      set(handles.fps1, 'Enable', 'off');
      set(handles.fps2, 'Enable', 'off');
      set(handles.fps3, 'Enable', 'off');
      set(handles.fps4, 'Enable', 'off');
      set(handles.fps5, 'Enable', 'off');
% Zároveň nastavím všechny radiobuttony na neoznačené
      set(handles.fps1, 'Value', 0);
      set(handles.fps2, 'Value', 0);
      set(handles.fps3, 'Value', 0);
      set(handles.fps4, 'Value', 0);
      set(handles.fps5, 'Value', 0);
% Nastavím String do colorCodingEdit
      set(handles.colorCodingEdit, 'String', 'Y (Mono 16)');
% Nastavím maximum posuvníku pro zisk
      set(handles.gainSlider, 'Max', 1023);
% Zpřístupním pole Packet Size
      set(handles.packetSizeEdit, 'Enable', 'on');
% Nastavím pole Packet Size na defaultní hodnotu
      set(handles.packetSizeEdit, 'String', '850');
% Nastavím hodnotu frame rate do infopanelu pro defaultní hodnotu
      set(handles.frameRateEdit, 'String', '9.412');
% Nastavím hodnotu maxima pro Gain Slider
      set(handles.gainSlider, 'Max', 1023);
```

Rozlišení 1024x768 pixelů, mono16:

```
      handles.kamera = 0;
      set(handles.widthEdit, 'String', '1024');
      set(handles.heightEdit, 'String', '768');
      handles.sirka = str2num(get(handles.widthEdit, 'String'));
      handles.vyska = str2num(get(handles.heightEdit, 'String'));
      set(handles.leftEdit, 'String', '0');
      set(handles.topEdit, 'String', '0');
% Dostupné fps pro dané rozlišení povolím, zbytek zakážu
      set(handles.fps1, 'Enable', 'off');
      set(handles.fps2, 'Enable', 'on');
      set(handles.fps3, 'Enable', 'on');
      set(handles.fps4, 'Enable', 'on');
      set(handles.fps5, 'Enable', 'off');
% Stejně, jak to dělá kamera, označím nejvyšší možný frame rate
      set(handles.fps4, 'Value', 1);
% Uložím do proměnné fps hodnotu označeného radiobuttonu
      handles.fps = '15';
      set(handles.colorCodingEdit, 'String', 'Y (Mono 16)');
% Změním maximum posuvníku pro zisk
      set(handles.gainSlider, 'Max', 511);
      set(handles.gainEdit, 'String', '370');
      set(handles.gainSlider, 'Value', 370);
% Nastavím packet size jako needitovatelný
      set(handles.packetSizeEdit, 'Enable', 'off');
% Zapíši příslušnou hodnotu hodnotu do packetSizeEdit, tato hodnota je
pouze informativní, protože toto rozlišení se řídí fps
      set(handles.packetSizeEdit, 'String', '3072');
```

Pro funkčnost aplikace jsem musel zjistit, které z dostupných 11 kombinací uživatel vybral, abych mohl nastavit výše uvedené vlastnosti.

```
% Do proměnné seznam načtu stringy, které jsou uloženy v rozbalovacím
menu Video menu
      seznam = get(handles.videoModeMenu,'String');
% Načtu pořadí vybraného stringu (1-11)
      handles.poradi = get(handles.videoModeMenu,'Value');
% Do proměnné rozlišení načtu string z daného pořadí
handles.rozliseni = seznam{handles.poradi};
```

Hodnoty v **edit text boxu** „Packet Size“ se liší podle rozlišení a označených fps. Zde je část kódu, kde definuji, jak se mají jednotlivé hodnoty nastavit a jaké fps uživatel vybral, pro vybraný **radio button**.

```
% Podle Tagu radiobuttonu, který je označen, rozhodnu, jaká sekvence
% příkazů se má vykonat
switch get(eventdata.NewValue,'Tag')
% Je-li vybrán 1. radiobutton, udělej následující sekvenci příkazů
    case 'fps1'
      .
      .
      .
% Je-li vybrán 5. radiobutton, udělej následující sekvenci příkazů
    case 'fps5'
      % Zapíši do proměnné fps příslušnou hodnotu
        handles.fps = '30';
      % Jestliže je hodnota pořadí (pořadí z video mode) rovna 5
        if handles.poradi == 5
            % Nastav Packet size na 2560
            set(handles.packetSizeEdit, 'String', '2560');
      % Jestliže je hodnota pořadí rovna 7
        elseif handles.poradi == 7
            % Nastav Packet size na 640
            set(handles.packetSizeEdit, 'String', '640');
      % Jestliže je hodnota pořadí rovna 10
        elseif handles.poradi == 10
            % Nastav Packet size na 1280
            set(handles.packetSizeEdit, 'String', '1280');
        end % Konec příkazu if
end % Konec příkazu switch
```

Pod panelem „Region of Interest“ se skrývá šest funkcí, které ho obsluhují. Jedná se o funkce `topEdit_Callback`, `leftEdit_Callback`, `heightEdit_Callback`, `widthEdit_Callback`, `manualSelectionButton_Callback` a `roiButton_Callback`. Funkce, které mají obsaženy v názvu „Edit“, se starají o funkčnost stejnojmenných **edit text boxů** v aplikaci a příkazy v roiButton callback se vykonají po tisknutí **push buttonu** „Reset to Defaults“. Funkce topEdit a heightEdit jsou spárované podle logiky ROI, stejně jako leftEdit a widthEdit. **Toggle button** „Manual Selection“ slouží k tomu, aby uživatel mohl při spuštěném náhledu vybrat ROI podle toho, co kamera právě snímá. Po jeho stisknutí může uživatel v náhledu tažením vytvořit obdélník, který lze zvětšovat, zmenšovat či posunovat. Plocha snímku zobrazená uvnitř toho obdélníku bude poté zvoleným ROI. Jsme-li spokojeni s vybranou oblastí, opětovně stikneme **toggle button** „Manual Selection“, čímž předáme hodnoty kameře. Následující zdrojový kód definuje funkce topEdit a heightEdit:

HeightEdit callback:

```
% Načtu string z pole heightEdit, převedu ho na číslo a porovnám jej
s proměnnou vyska, což je výška rozlišení
if str2num(get(handles.heightEdit, 'String')) > handles.vyska
      % Je-li hodnota v heightEdit větší, nastavím string v heightEdit
      zpětně na maximální hodnotu, kterou představuje vyska
    set(handles.heightEdit, 'String', num2str(handles.vyska));
      % Do proměnné height uložím hodnotu heightEdit jako číslo
    handles.height = str2num(get(handles.heightEdit, 'String'));
      % Zadá-li uživatel číslo menší než 0
elseif str2num(get(handles.heightEdit, 'String')) < 0
      % Nastaví se hodnota v textovém poli Height na 0
    set(handles.heightEdit, 'String', '0');
      % Do proměnné height uložím hodnotu heightEdit jako číslo
    handles.height = str2num(get(handles.heightEdit, 'String'));
% V jiném případě (je-li heightEdit <= vyska, ale vyšší než 0) pouze uložím
hodnotu z pole heightEdit jako číslo
else handles.height = str2num(get(handles.heightEdit, 'String'));
end

% Zde je ošetřen případ, když by uživatel zadal do textového pole Height
hodnotu tak velkou, že součet hodnot z polí Top a Height je větší než
výška rozlišení, nastaví se do pole Top hodnota odpovídající
rozdílu celkové výšky rozlišení a zadané hodnoty Height (viz obr. 22)
if handles.ROItop + handles.height > handles.vyska
   set(handles.topEdit, 'String', num2str(handles.vyska - handles.height));
end
guidata(hObject, handles);
```

Obr. 22 Princip fungování ROI v heightEdit callbacku pro rozlišení 800x600 pixelů

TopEdit callback:

```
% Tento cyklus if je v podstatě totožný jako u heightEdit callbacku
if str2num(get(handles.topEdit, 'String')) > handles.vyska
    set(handles.topEdit, 'String', num2str(handles.vyska));
    handles.ROItop = str2num(get(handles.topEdit, 'String'));
elseif str2num(get(handles.topEdit, 'String')) < 0
    set(handles.topEdit, 'String', '0');
    handles.ROItop = str2num(get(handles.topEdit, 'String'));
else handles.ROItop = str2num(get(handles.topEdit, 'String'));
end
% Zde je výpočet výšky zobrazené oblasti neboli hodnota rozdílu výšky
rozlišení a čísla zadaného uživatelem do pole Top
handles.ROIheight = handles.vyska - handles.ROItop;
% Nastavím do textového pole Height vypočtenou výšku, ale pouze v případě,
je-li sočet čísel z Top a Hieght vyyší než je celkové rozlišení, jinak se
hodnota v Height nepřepisuje (viz obr. 23)
if handles.height + handles.ROItop > handles.vyska
    set(handles.heightEdit, 'String', num2str(handles.ROIheight));
end
```

Obr. 23 Princip fungování ROI v topEdit callbacku pro rozlišení 800x600 pixelů

Tento zdrojový kód obsahuje funkcionalitu **push buttonu** „Reset to Defaults“:

```
% Nastavím textové pole Left a Top na defaultní hodnotu 0
      set(handles.leftEdit, 'String', '0');
      set(handles.topEdit, 'String', '0');
% Nastavím textové pole Width na defaultní hodnotu šířka rozlišení
      set(handles.widthEdit, 'String', num2str(handles.sirka));
% Nastavím textové pole Height na defaultní hodnotu výška rozlišení
      set(handles.heightEdit, 'String', num2str(handles.vyska));
```

Následující kód aplikace popisuje, jak funguje **toggle button** „Manual Selection“:

```
% Zjistím, v jakém stavu je toggle button
      button_state = get(hObject,'Value');
% Je-li zmáčknutý, uživatel může vybrat ROI
      if button_state == get(hObject,'Max')
            handles.manualROI = imrect(handles.axes1);
```

```
      % Je-li nezmáčknutý
      elseif button_state == get(hObject,'Min')
            % Zjistím souřadnice vybraného ROI
            pos = getPosition(handles.manualROI);
            % Načtu je do jednotlivých proměnných
            manualROIleft = round(pos(1));
            manualROItop = round(pos(2));
            manualROIwidth = round(pos(3));
            manualROIheight = round(pos(4));
            set(handles.leftEdit, 'String', num2str(manualROIleft));
            % Příslušné textboxy nastavím na zjištěné hodnoty
            set(handles.topEdit, 'String', num2str(manualROItop));
            set(handles.widthEdit, 'String', num2str(manualROIwidth));
            set(handles.heightEdit, 'String', num2str(manualROIheight));
            set(handles.manualROI, 'Visible', 'off');
end
```

4.3.3 Atributy kamery

Callbacky pro jednotlivé atributy kamery jsou velmi podobné. Musel jsem zohledňovat minima, maxima a defaultní hodnoty. U zisku jsem navíc musel brát v úvahu, jaké rozlišení je právě vybráno. Pro každý atribut jsem vytvořil tři funkce. Jedna se stará o **radio buttony** a zbylé dvě o samotné propojení **edit text boxu** a **slideru**.

Nejdříve testuji, zda uživatel vybral u panelu „Gain“ **radio button** „Manual“ nebo „Default“:

```
% Podle Tagu označeného radiobuttonu, rozhodnu, jaké příkazy se vykonají
switch get(eventdata.NewValue,'Tag')
      % Je-li vybrán Manual, zpřístupním textové pole i slider
    case 'manualGain'
        set(handles.gainEdit, 'Enable', 'on');
        set(handles.gainSlider, 'Enable', 'on');
      % Je-li vybrán Default, znepřístupním textové pole a slider a
      nastavím na obou komponentách defaultní hodnotu pro zisk
    case 'ignoreGain'
        set(handles.gainEdit, 'String', '370');
        set(handles.gainSlider, 'Value', 370);
        set(handles.gainEdit, 'Enable', 'off');
        set(handles.gainSlider, 'Enable', 'off');
end % Konec Switch
```

Jestliže byl označen „Manual“, mohu na **slideru** nastavit hodnotu, která se okamžitě objevuje v **edit text boxu** a je zaokrouhlena na celé číslo:

```
% Načtu a zaokrouhlím hodnotu ze Sliderů zisku
      roundGain = round(get(handles.gainSlider,'Value'));
% Zaokrouhlenou hodnotu převedu na string
      gainSliderValue = num2str(roundGain);
% Nastavím získaný string do pole gainEdit
      set(handles.gainEdit, 'String', gainSliderValue);
```

Obdobný, ale opačný proces platí u funkce pro **edit text box**. Zde jsem navíc musel ošetřit, aby uživatel nemohl zadat menší nebo větší hodnotu než je přípustná, protože by kamera nemohla správně pracovat:

```
% Načtu hodnotu z pole gainEdit a převedu ji na číslo
      gainEditValue = str2double(get(handles.gainEdit, 'String'));
% Jestliže je hodnota větší než 1023 nebo 511 (maximální hodnota), nastavím
textové pole gainEdit a gainSlider na maximum
      if  gainEditValue > get(handles.gainSlider, 'Max')
            set(handles.gainEdit, 'String', '1023');
            set(handles.gainSlider, 'Value', 1023);
% Jestliže je hodnota menší než 350 (minimální hodnota), nastavím pole
gainEdit i gainSlider na minimum
      elseif gainEditValue < 350
            set(handles.gainEdit, 'String', '350');
            set(handles.gainSlider, 'Value', 350);
% Ve zbývajícím případě nastavím gainSlider na hodnotu z textového pole
      else
            set(handles.gainSlider, 'Value', gainEditValue);
      end % Konec if
```

4.3.4 Push buttony Start/Stop Preview

Push buttony „Start“ nebo „Stop Preview“ umožňují spustit nebo zastavit náhled kamery. Uživatel tak zjistí, jaké atributy musí ještě změnit, aby dosáhl požadovaného nastavení snímku.

Nejdříve musím vytvořit video objekt s rozlišením, které bylo vybráno, pokud již neexistuje:

```
if handles.kamera == 0
handles.vid = videoinput('dcam', 1, handles.rozliseni);
handles.src = getselectedsource(handles.vid);
handles.kamera = 1;
end
```

Poté postupně načítám všechny vlastnosti a hodnoty, které kamera umožňuje nastavit a které potřebuje k správnému fungování. V další fázi je předávám kameře. Nakonec spustím náhled a do konzole vypíšu, že náhled byl spuštěn:

```
% Načtu hodnotu pro Brightness
 brightness = str2double(get(handles.brightnessEdit, 'String'));
% Načtu hodnotu pro Gain
 gain = str2double(get(handles.gainEdit, 'String'));
% Načtu hodnotu pro Shutter
 shutter = str2double(get(handles.shutterEdit, 'String'));
% Nastavím kameře Brightness
 set(handles.src, 'Brightness', brightness);
% Nastavím kameře Gain
 set(handles.src, 'Gain', gain);
% Nastavím kameře Shutter
 set(handles.src, 'Shutter', shutter);
% Nastavím kameře FrameRate pokud jej má (konkrétně pro všechny rozlišení
kromě 1 a 2)
 if handles.poradi >= 3
 set(handles.src, 'FrameRate', handles.fps);
 end
```

```
% Načtu hodnoty pro Region of Interest
 ROIleft = str2num(get(handles.leftEdit, 'String'));
 ROItop = str2num(get(handles.topEdit, 'String'));
 ROIwidth = str2num(get(handles.widthEdit, 'String'));
 ROIheight = str2num(get(handles.heightEdit, 'String'));
% Nastavím ROI zvolené uživatelem
 set(handles.vid, 'ROIPosition', [ROIleft ROItop ROIwidth ROIheight]);
```

Dále nastavuji vlastnosti `handles.axes1` tak, aby náhled byl zobrazen právě v objektu „axes1“ a aby byl vždy ve středu zobrazované plochy:

```
% Sekvence příkazů, která zajišťuje, aby se náhled kamery zobrazil v axes1
 vidRes = get(handles.vid, 'VideoResolution');
 imWidth = vidRes(1);
 imHeight = vidRes(2);
 nBands = get(handles.vid, 'NumberOfBands');
 hImage = image(zeros(imHeight, imWidth, nBands), 'parent', handles.axes1);
% Nastavím axes1 jako viditelný
 set(handles.axes1, 'Visible', 'on');
% Blok výpočtů pro umístění axes vždy do středu náhledové plochy. Vzhledem
k tomu, že náhled nemá skutečnou velikost rozlišení, použil jsem poměrové
vyjádření.
 roi_axes_x = (ROIwidth / handles.sirka) * handles.axes_sirka;
 roi_axes_y = (ROIheight / handles.vyska) * handles.axes_vyska;
% Výpočet velikosti výsledného odsazení zleva (x) a z horní části (y)
 odsazeni_x = (handles.axes_sirka - roi_axes_x) / 2;
 odsazeni_y = (handles.axes_vyska - roi_axes_y) / 2;
% Výpočet počátečních souřadnit náhledového okna
 roi_x = handles.axes_x + odsazeni_x;
 roi_y = handles.axes_y - odsazeni_y;
% Nastavím náhledovému axes1 vypočítané hodnoty v jednotkách pixel
 set(handles.axes1, 'Position', [roi_x roi_y handles.axes_sirka...
                                 handles.axes_vyska]);
% Jednotky nastavím na „normalized“, tím se stanou univerzální pro všechny
typy monitorů a mohu je využívat pro následující výpočty
 set(handles.axes1, 'Units', 'Normalized');
```

Uživateli ukážu, co může při zobrazení náhledu upravovat:

```
% Zpřístupnění/znepřístupnění tlačítek
 set(handles.startPreviewButton, 'Enable', 'off');
 set(handles.stopPreviewButton, 'Enable', 'on');
 set(handles.videoModeMenu, 'Enable', 'off');
 set(handles.framesPerTrigger, 'Enable', 'off');
 set(handles.triggerCount, 'Enable', 'off');
```

Nakonec spustím náhled a do konzole vypíšu, že náhled byl spuštěn.

```
% Spustím náhled
 preview(handles.vid, hImage);
% Výpis do konzole
 disp('Preview started');
```

Push button „Stop Preview“ skrývá několik funkcí. Náhled zastaví a vypíše do konzole, že byl zastaven. Převede jednotky objektu `handles.axes1` znovu na pixely, abych mohl později opět

vypočítat jeho umístění do středu plochy, a zpřístupní veškeré komponenty, které jsem předtím zakázal:

```
% Zpřístupnění/znepřístupnění tlačítek
set(handles.startPreviewButton, 'Enable', 'on');
set(handles.stopPreviewButton, 'Enable', 'off');
set(handles.videoModeMenu, 'Enable', 'on');
set(handles.framesPerTrigger, 'Enable', 'on');
set(handles.triggerCount, 'Enable', 'on');
% Po zastavení náhledu musím jednotky od axes znovu nastavit na pixely,
% abych mohl provádět výpočty
set(handles.axes1, 'Units', 'Pixels');

% Zastavím Preview
stoppreview(handles.vid);

% Výpis do konzole
disp('Preview stopped');
```

4.3.5 Push buttony Start/Stop Acquisition

Push button „Start Acquisition“ umožňuje spustit snímání kamery. Sekvence příkazů je podobná jako u spuštění náhledu, ale předávám kameře další hodnoty, které u náhledu nejsou zapotřebí. Jedná se o trigger, počet framů a počet triggerů:

```
% Nastavím manuální trigger
 triggerconfig(handles.vid, 'manual');

% Načtu počet framů
 handles.pocetFrames = str2double(get(handles.framesPerTrigger, 'String'));
% Načtu počet triggerů a odečtu 1
 handles.pocetTriggers = str2double(get(handles.triggerCount, 'String'))-1;
% Nastavím kameře počet Framů
 set(handles.vid, 'FramesPerTrigger', handles.pocetFrames);
% Nastavím kameře počet Triggerů
 set(handles.vid, 'TriggerRepeat', handles.pocetTriggers);
```

Zpřístupním potřebné **push buttony** a zakážu měnit všechny vlastnosti, protože to již u snímání není možné. Tato část kódu, kde zakazuji používat téměř všechny komponenty je velmi dlouhá, proto jako ukázku použiji pouze změnu stavu **push buttonů**:

```
 % Zpřístupnění/znepřístupnění tlačítek
 set(handles.startPreviewButton, 'Enable', 'off');
 set(handles.startAcquisitionButton, 'Enable', 'off');
 set(handles.triggerButton, 'Enable', 'on');
 set(handles.stopAcquisitionButton, 'Enable', 'on');
```

Poté spustím náhled a proces snímání u kamery, abychom mohli pořídit snímky:

```
 % Spustím Preview
 preview(handles.vid, hImage);
 % Spustím kameru
 start(handles.vid);
 % Výpis do konzole
 disp('Kamera start');
```

Nakonec pomocí následující sekvence příkazů měním jednotlivé položky v panelu „Information Panel“:

```
% Podle toho, jestli má nebo nemá kamera v paměti data,
% výpíše se do informačního panelu její stav (Yes/No)
if islogging(handles.vid) == 1
    set(handles.camLogEdit, 'String', 'Yes');
else set(handles.camLogEdit, 'String', 'No');
end

% Podle toho, jestli je kamera spuštěna a připravena pro snímání,
% výpíše se do informačního panelu její stav (Yes/No)
if isrunning(handles.vid) == 1
    set(handles.camRunEdit, 'String', 'Yes');
else set(handles.camRunEdit, 'String', 'No');
end

% Do informačního panelu se vypíše kolik framů je k dispozici pro uložení
handles.framesAvailable = get(handles.vid, 'FramesAvailable');
set(handles.framesAvailableEdit, 'String', handles.framesAvailable);

% Do informačního panelu se vypíše kolik framů bylo nasnímáno
framesAcquired = get(handles.vid, 'FramesAcquired');
set(handles.framesAcqEdit, 'String', framesAcquired);
```

Příkazy skryté pod **push buttonem** „Stop Acquisition“ jsou také velmi podobné jako u „Stop Preview“. Zastavím náhled a proces snímání, převedu jednotky objektu `handles.axes1` na pixely a zpřístupním všechny komponenty, které jsem dříve zakázal. Nejpodstatnější je, že umožním uložit data tím, že zpřístupním **push button** „Save Data“.

```
% Zastavím Preview
stoppreview(handles.vid);
% Zastavím kameru
stop(handles.vid);
set(handles.axes1, 'Units', 'Pixels');
% Zpřístupnění/znepřístupnění tlačítek
set(handles.startPreviewButton, 'Enable', 'on');
set(handles.startAcquisitionButton, 'Enable', 'on');
set(handles.triggerButton, 'Enable', 'off');
set(handles.stopAcquisitionButton, 'Enable', 'off');
% Zpřístupnění tlačítko save data, jsou-li nějaká data nasnímána
if get(handles.vid, 'FramesAvailable') >= 1
    set(handles.saveDataButton, 'Enable', 'on');
end

% Výpis do konzole
disp('Kamera stop');

% Výpisy pro informační panel
if islogging(handles.vid) == 1
    set(handles.camLogEdit, 'String', 'Yes');
else set(handles.camLogEdit, 'String', 'No');
end
```

```
if isrunning(handles.vid) == 1
    set(handles.camRunEdit, 'String', 'Yes');
else set(handles.camRunEdit, 'String', 'No');
end

framesAvailable = get(handles.vid, 'FramesAvailable');
set(handles.framesAvailableEdit, 'String', framesAvailable);

framesAcquired = get(handles.vid, 'FramesAcquired');
set(handles.framesAcqEdit, 'String', framesAcquired);
```

4.3.6 Push buttony Trigger a Save Data

Push button „Trigger“ slouží jako tzv. spouštěč. Při kliknutí na „Trigger“ se do paměti kamery uloží předem nastavený počet framů a do konzole se vypíše, že byl proveden trigger. Počet triggerů závisí na čísle v **edit text boxu** „Number of Triggers“. Proces snímání se neustále opakuje, dokud se nevyčerpá počet triggerů, který byl nastaven před snímáním. Jakmile je tato hodnota rovna nule, dojde k zastavení náhledu a zpřístupnění **push buttonu** „Stop Acquisition“. Zároveň se hodnota nastaví zpět na číslo 1, aby mohla aplikace dále pokračovat při dalším snímání. I v této funkci je sekvence příkazů pro panel „Information Panel“, ale nebudu je znovu uvádět, jelikož se nemění:

```
% Načtu počet triggerů
i = str2num(get(handles.triggerCount, 'String'));

% Vyfotím zobrazený snímek
trigger(handles.vid);

% Snížím počet triggerů o 1, protože byl 1 proveden
i = i - 1;
% Nastavím textové pole Number of Triggers na výslednou hodnotu. Uživatel
% tak vidí, kolik triggerů má ještě k dispozici.
set(handles.triggerCount, 'String', num2str(i));

% Dojde-li k vyčerpání všech triggerů
if i < 1
    % Zastavím náhled
    stoppreview(handles.vid);
    % Zpřístupnění/znepřístupnění tlačítek
    set(handles.startPreviewButton, 'Enable', 'off');
    set(handles.startAcquisitionButton, 'Enable', 'off');
    set(handles.saveDataButton, 'Enable', 'off');
    set(handles.triggerButton, 'Enable', 'off');
    set(handles.stopAcquisitionButton, 'Enable', 'on');
    % Aby uživatel nemusel následně přepisovat pole Number of Triggers,
    % nastavím na defaultní číslo 1. Zároveň tím docílím fungování aplikace
    % při následném spuštění
    set(handles.triggerCount, 'String', '1');
end

% Výpis do konzole
disp('Trigger done');
```

Push button pro ukládání snímků slouží pro načtení všech vyfocených snímků. Po načtení snímků z paměti kamery je nutné tyto snímky pojmenovat a uložit do počítače. Proto nejdříve testuji, kolik framů bylo pořízeno. Jestliže byl pořízen pouze jeden snímek, bude možné jej uložit ve formátech *.jpg, *.tiff a *.png. Jestliže paměť kamery obsahuje více než jeden snímek, bude je uživatel postupně ukládat po jednom snímku. Je sice možné ukládat více snímků najednou v dostupných videoformátech, ale tyto videoformáty jsou kompresní, a proto tato varianta není vhodná. Počet snímků testuji vynásobením hodnoty v **edit text boxech** „Frames per Trigger“ a „Number of Triggers“. Tyto instrukce se provádějí v pozadí aplikace a uživatel vidí pouze obyčejný průzkumník pro ukládání dat, viz obr. 21. To umožňuje funkce uiputfile, pomocí které mohu nastavit, jaké formáty bude mít uživatel k dispozici:

```
% Celkový počet framů, které byly snímány (počet framů * počet triggerů)
celkemFramu = handles.pocetFrames * (handles.pocetTriggers + 1);

% Jestliže je vyfocen pouze 1 snímek
if celkemFramu == 1
      % Načtení vyfoceného snímku
      snimek = getdata(handles.vid);
      % Do proměnné uložím časový formát 30 (yyyymmddTHHMMSS), který
      použiju jako jméno souboru a pomocí funkce strcat vytvořím defaultní
      název souboru ve formátu png
      jmeno = datestr(now,30);
      celeJmeno = strcat(jmeno, '.png');
      Pomocí funkce uiputfile nastavím, z jakých formátů souborů může
      uživatel vybírat, a načtu cestu uložení souboru a jeho jméno
      [file,path] = uiputfile({'*.png', 'PNG format(*.png)';...
            '*.jpg', 'JPG format(*.jpg)';...
            '*.tif', 'TIF format(*.tif)';...
            '*jpg;*.png;*.tif;','All Image Files (*.jpg;*.png;*.tif)';...
            '*.*','All Files (*.*)' },...
            'Save Image', celeJmeno);
      % Pomocí funkce fullfile vytvořím kompletní cestu k souboru
      fullPath = fullfile(path, file);
      % Jestliže uživatel kliknul na Cancel (nebyla tedy zadána žádna cesta
      pro uložení), snímky budou vymazány. Zobrazí se varovný dialog.
      if path == 0
        warndlg('Save cancelled, frame(s) deleted!')
      % V opačném případě testuji, který formát uživatel vybral pro uložení
      else
            % Jestliže se jedná o JPG formát, uložím pod vytvořeným jménem
            a cestou a vypíši na obrazovku, že uložení proběhlo úspěšně. O
            jaký formát se jedná, testuji pomocí funkce strfind.
            if isempty(strfind(file, '.jpg'))== 0
            imwrite(snimek, fullPath,'Bitdepth', 16);
            warndlg('Save successfull.');
            % PNG formát
            elseif isempty(strfind(file, '.png'))== 0
            imwrite(snimek, fullPath);
            warndlg('Save successfull.');
            % TIF formát
            elseif isempty(strfind(file, '.tif'))== 0
            imwrite(snimek, fullPath);
            warndlg('Save successfull.');
```

```
            % V jiném případě se snímek sice uloží, ale bez formátu
            (přípony)
            else
            imwrite(snimek, fullPath);
            warndlg('Save failed! Unknown format.')
        end
    end
% Jestliže bylo pořízeno více snímků
else
for i = 1:celkemFramu

    % Načtení jednoho vyfocených snímků
      snimky = getdata(handles.vid, 1);
      % Do proměnné uložím časový formát 30 (yyyymmddTHHMMSS), který
      použiju jako jméno souboru a pomocí funkce strcat vytvořím defaultní
      název souboru ve formátu png
      jmeno = datestr(now,30);
      cislo = sprintf('_%d.png', i);
      celeJmeno = strcat(jmeno, cislo);
      Pomocí funkce uiputfile nastavím, z jakých formátů souborů může
      uživatel vybírat, a načtu cestu uložení souboru a jeho jméno
      [file,path] = uiputfile({'*.png', 'PNG format(*.png)';...
            '*.jpg', 'JPG format(*.jpg)';...
            '*.tif', 'TIF format(*.tif)';...
            '*jpg;*.png;*.tif;','All Image Files (*.jpg;*.png;*.tif)';...
            '*.*','All Files (*.*)' },...
            'Save Image', celeJmeno);
      % Pomocí funkce fullfile vytvořím kompletní cestu k souboru
      fullPath = fullfile(path, file);
      % Jestliže uživatel kliknul na Cancel (nebyla tedy zadána žádna cesta
      pro uložení), snímky budou vymazány. Zobrazí se varovný dialog.
      if path == 0
        warndlg('Save cancelled, frame(s) deleted!')
      % V opačném případě testuji, který formát uživatel vybral pro uložení
      else
            % Jestliže se jedná o JPG formát, uložím pod vytvořeným jménem
            a cestou a vypíši na obrazovku, že uložení proběhlo úspěšně. O
            jaký formát se jedná, testuji pomocí funkce strfind.
            if isempty(strfind(file, '.jpg'))== 0
            imwrite(snimek, fullPath,'Bitdepth', 16);
            warndlg('Save successfull.');
            elseif isempty(strfind(file, '.png'))== 0 % PNG formát
            imwrite(snimek, fullPath);
            warndlg('Save successfull.');
            elseif isempty(strfind(file, '.tif'))== 0% TIF formát
            imwrite(snimek, fullPath);
            warndlg('Save successfull.');
            % V jiném případě se snímek sice uloží, ale bez formátu
            (přípony)
            else
            imwrite(snimek, fullPath);
            warndlg('Save failed! Unknown format.')
            end
      end
      framesAvailable = get(handles.vid, 'FramesAvailable');
      set(handles.framesAvailableEdit, 'String', framesAvailable);
end
end
```

Závěr

Cílem této bakalářské práce bylo vytvořit plně funkční uživatelské prostředí, které by komunikovalo s kamerou Basler A631f, na základě předlohy GUI vytvořené v systému MAX. K tvorbě jsem využil sadu nástrojů GUIDE, který je součástí MATLABu, protože jsem potřeboval docílit uceleného a intuitivního vzhledu aplikace, která musí být minimálně omezena rozlišením monitoru, na kterém bude spuštěna. Těchto vlastností bych ručním programováním vzhledu GUI nedosáhl. Při vytváření aplikace jsem dodržoval zavedená pravidla pro tvorbu GUI a dle mého názoru jsem dokázal vylepšit strukturu GUI oproti předloze.

Zdrojový kód aplikace je výstižně komentovaný a psaný tak, aby bylo možné ho v budoucnu jakkoli rozšířit nebo modifikovat. Taktéž vzhled aplikace je rozdělen do jednotlivých panelů, které se v budoucích verzích MATLABu budou moci sloučit, až bude podporován systém záložek s dostatečnou funkcionalitou. Dojde tak k ušetření místa na pracovní ploše aplikace, náhledové okno bude moci být větší a tudíž bude mít uživatel k dispozici kvalitnější náhled. V případě napojení jiné kamery bude GUI i nadále fungovat a po lehké úpravě vzhledu plně využije její možnosti.

Samotná textová část mé bakalářské práce obsahuje detailní popis všech vlastností kamery, jejího propojení s PC a bližší popis programů a jejich komponent, se kterými pracuji. Dále je v ní uveden podrobný rozbor vzhledu mého GUI a GUI ze systému MAX. Poslední část tvoří manuál pro práci s vytvořenou aplikací a stručný popis zdrojového kódu aplikace, ve kterém jsem se snažil popsat a co nejlépe vysvětlit nejzajímavější a nejkomplikovanější sekvence příkazů.

Použitá literatura

[1] Basler A630f. *Motion Analysis Inc.* [online]. 29.6.2005 [cit. 2013-04-16]. Dostupné z: http://www.motionanalysisinc.com/specs/basler/a631fman.pdf

[2] Image Acquisition Toolbox: Supported Hardware. *The MathWorks, Inc.* [online]. © 1994-2013 [cit. 2013-04-16]. Dostupné z: https://www.mathworks.com/products/imaq/supported/dcam-compatible-firewire-cameras.html

[3] Installation. *CMU 1394 Digital Camera Driver* [online]. 26. 9. 2011 [cit. 2013-04-16]. Dostupné z: http://www.cs.cmu.edu/~iwan/1394/install.html

[4] MATLAB. *HUMUSOFT* [online]. © 1991 - 2013 [cit. 2013-04-16]. Dostupné z: http://www.humusoft.cz/produkty/matlab/matlab/

[5] Image Acquisition Toolbox 3. *Melafrit* [online]. © 2007 [cit. 2013-04-16]. Dostupné z: http://www.melafrit.com/education/ENSEIRB/Matlab/MatlabTutoriel/91109v04.pdf

[6] Image Acquisition Toolbox™. *HUMUSOFT* [online]. © 1991 - 2013 [cit. 2013-04-16]. Dostupné z: http://www.humusoft.cz/produkty/matlab/aknihovny/imaq/

[7] Kovářík, Martin. *Programování a tvorba grafiky v MATLABu.* Vydání 1. Zlín: Univerzita Tomáše Bati ve Zlíně, 2008. 130 s. ISBN 978-80-7318-754-5 (brož.).

[8] Graphical User Interface. *ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE* [online]. [2005] [cit. 2013-04-16]. Dostupné z: https://cw.felk.cvut.cz/lib/exe/fetch.php/courses/y36pjv/s08_gui.pdf

[9] ZAPLATÍLEK, K.; DOŇAR, B. *MATLAB: tvorba uživatelských aplikací.* Vydání 1. Praha: BEN - technická literatura, 2004. 215 s. ISBN 80-7300-133-0.

[10] MATELA, Lukáš. *Počítačová analýza obrazu úzkých textilií z hlediska jakosti zpracování a vad* [online]. Liberec, 2006 [cit. 2013-04-16]. Dostupné z: http://www.fm.tul.cz/files/autoreferat_matela.pdf. Autoreferát disertační práce. Technická univerzita v Liberci.

Seznam příloh

Adresářová struktura s přílohami na přiloženém CD

/bakalarska_prace/zadani_bakalarske_prace.pdf – zadání bakalářské práce

/bakalarska_prace/text_bakalarske_prace_vol0015.pdf - text bakalářské práce

/MATLAB/GUI.m – soubor MATLABu se zdrojovým kódem aplikace

/MATLAB/GUI.fig – soubor MATLABu s grafickou podobou aplikace